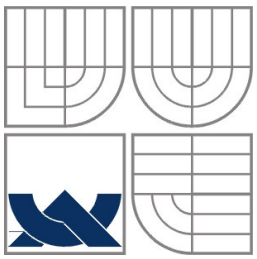
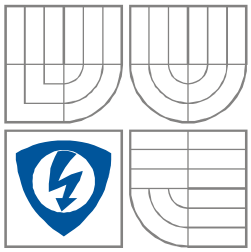


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

VŠESMĚROVÝ MOBILNÍ ROBOT

OMNIDIRECTIONAL MOBILE ROBOT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — ONDŘEJ ŠTĚRBA
AUTHOR

VEDOUCÍ PRÁCE — prof. Ing. FRANTIŠEK ZEZULKA, CSc.
SUPERVISOR

BRNO 2012

VYSOKÉ UČENÍ
TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

Ústav automatizace a měřicí techniky

Bakalářská práce

bakalářský studijní obor
Automatizační a měřicí technika

Student: Ondřej Štěrba
Ročník: 3

ID: 125125
Akademický rok: 2011/12

NÁZEV TÉMATU:

Všesměrový mobilní robot

POKYNY PRO VYPRACOVÁNÍ:

Za pomocí univerzálních motorů vytvořte model všesměrového mobilního robotu. Navrhněte a realizujte spojení mobilního robotu s PC. Pro ovládání robotu vytvořte jednoduchý program pro PC, který bude zajišťovat ovládání robotu a vizualizaci dat získaných ze senzorů motorů.
Program doplňte o možnost plánování trajektorie pohybu. Navrhněte osazení robotu vhodnými senzory pro detekci překážek.

DOPORUČENÁ LITERATURA:

Robin R. Murphy: Introduction To Ai Robotics, The MIT press, 2000,
ISBN: 0-262-13383-0

Termín zadání: 6.2.2012 ***Termín odevzdání:*** 28.5.2012

Vedoucí práce: prof. Ing. František Zezulka, CSc.
Konzultanti bakalářské práce: Ing. Jaroslav Šembera

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ Fakulta elektrotechniky a komunikačních technologií
Ústav automatizace a měřicí techniky

UPOZORNĚNÍ:

Autor bakalářské práce nesmí při vytváření bakalářské práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Abstrakt

Práce se zabývá řízením robotického podvozku na bázi všesměrových („švédských“) kol, tzn. řízením úhlových rychlostí jednotlivých kol za účelem dosažení pohybu robotu žádaným směrem a žádanou rychlostí. Dále se práce zabývá detekcí překážek před robotem pomocí přídavných senzorů.

Klíčová slova

Robotický podvozek, Všesměrová (švédská) kola, Matlab Simulink, Servomotor Dynamixel RX-64, RS485, Joystick, DirectX, DirectInput, Senzory vzdálenosti, Atmel AVR, ATmega8

Abstract

This bachelor’s thesis describes controlling of omnidirectional chassis based on omnidirectional (‚swedish“) wheels especially controlling angular velocities for each wheel to achieve desired speed and desired direction. As next describes detecting obstacles by additional sensors.

Keywords

Robot chassis, Omnidirectional (swedish) wheels, Matlab Simulink, Actuator Dynamixel RX-64, RS485, Joystick, DirectX, DirectInput, Distance sensors, Atmel AVR, ATmega8

Bibliografická citace:

ŠTĚRBA, O. *Všesměrový mobilní robot*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2012, 52s. Vedoucí bakalářské práce prof. Ing. František Zezulka, CSc.

Prohlášení

„Prohlašuji, že svou bakalářskou práci na téma Všesměrový mobilní robot jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.
Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: **23. května 2012** ……
podpis autora

Obsah

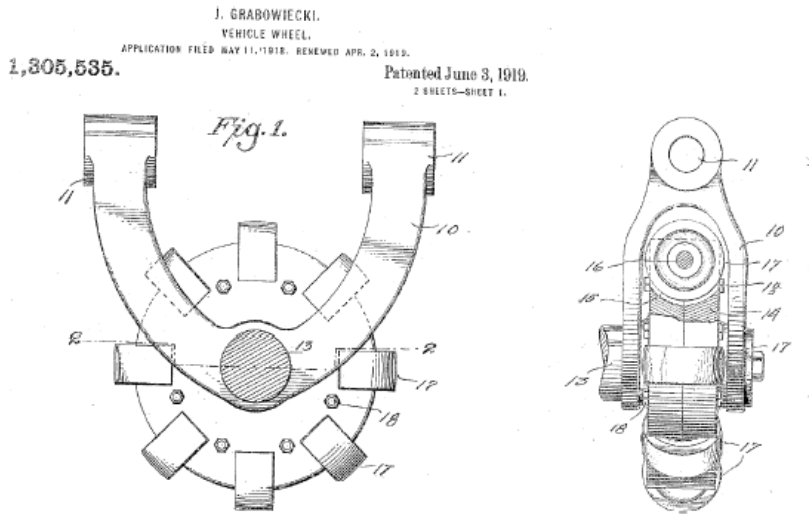
Seznam obrázků

Seznam tabulek

1 ÚVOD

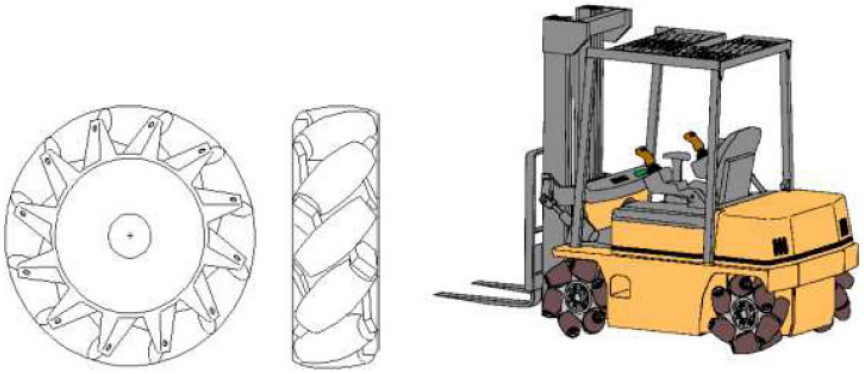
1.1 Historie všesměrových kol

Historie všesměrových kol sahá až do roku 1919, kdy J.Grabowiecki na tento druh kol podal patentovou přihlášku.


J. GRABOWIECKI.
VEHICLE WHEEL.
APPLICATION FILED MAY 11, 1918. RENEWED APR. 2, 1919.
1,305,535. Patented June 3, 1919.
2 SHEETS—SHEET 1.
Fig. 1.

Obrázek 1 Patentová příhláška z roku 1919

Nicméně ve své době se tento vynález příliš neuplatnil a teprve až kolem roku 1973 švédský konstruktér Berngt Ilon sestrojil první vysokozvižný vozík na bázi všesměrového podvozku, proto se tato kola někdy nazývají i jako „švédská kola“. Nicméně on použil poněkud modifikovanou konstrukci kol- válečky nebyly umístěny kolmo na osu otáčení kola, ale pod úhlem 45 stupňů.

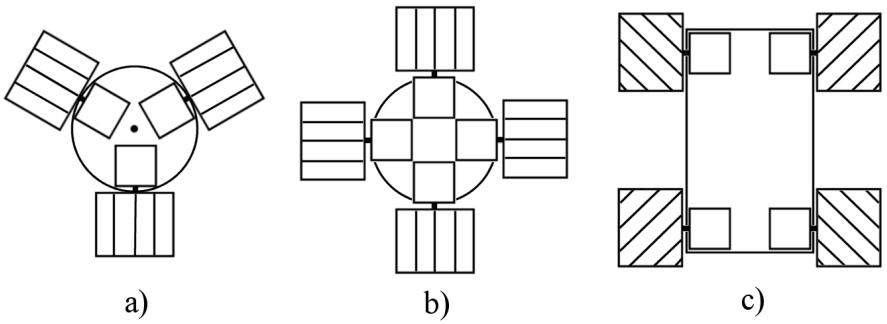


Obrázek 2 Vysokozdvižný vozík se všesměrovými koly

Existují tedy dvě hlavní varianty všesměrových (švédských) kol:

- S válečky umístěnými kolmo k ose otáčení kola
- S válečky umístěnými pod úhlem 45 stupňů

Při použití kol s kolmými válečky se obvykle volí rozmístění kol do kruhu (většinou 3 nebo 4 kola, ale lze použít i libovolný vyšší počet kol), v případě použití kol se šikmo umístěnými válečky se obvykle volí klasické umístění kol, tzn. dvě kola na levé straně a dvě kola na pravé straně.



Obrázek 3 Možná uspořádání podvozku se všesměrovými koly

1.2 Manévrovací schopnosti

Obecně každý druh podvozku má jiné manévrovací schopnosti, které jsou dány typem kol a rozmístěním kol. Každé kolo umožňuje robotu pohyb, ale zároveň každé kolo obvykle přináší i omezení pohybu (např. klasické kolo umožňuje robotu pohyb v jedné ose, ale zároveň v druhé kolmé ose pohybu zabraňuje).

Tyto manévrovací schopnosti lze popsat pomocí parametetrů **mobilita, řiditelnost a manévrovatelnost** (těmito pojmy se více podrobněji zabývá literatura [14])

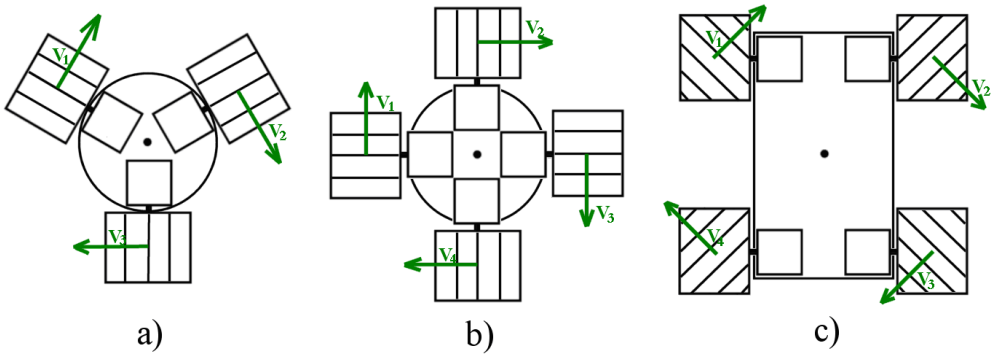
Stupeň mobility je závislý na tom, u kolika kol můžeme nezávisle měnit rychlost otáčení nebo jinými slovy pomocí kolika parametrů můžeme jednoznačně určit rychlosti otáčení všech kol. Na počtu kol přitom až tolik nezáleží (například u bicyklu můžeme rychlost pohybu jednoznačně určit jedním parametrem i když bicykl má fyzicky kola dvě)

Stupeň řiditelnosti je závislý na tom, kolika koly můžeme nezávisle na sobě zatáčet, neboli pomocí kolika parametrů můžeme jednoznačně určit natočení všech řiditelných kol, na počtu kol přitom opět nezáleží (například i když u některých autobusů mírně

zatáčí i zadní náprava, zůstává stupeň řiditelnosti 1, protože mezi natočením všech kol existuje lineární závislost)

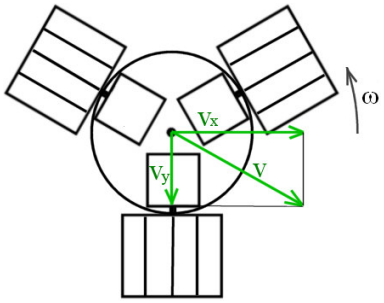
Stupeň manévrovatelnosti je definován jako součet stupně mobility a stupně řiditelnosti.

U tříkolového všesměrového podvozku (nezávisle na použité variantě) můžeme určovat rychlost každého kola nezávisle na ostatních, takže stupeň mobility je 3. Čtyřkolový všesměrový podvozek má stupeň mobility také 3, protože tam už mezi koly existuje určitá lineární závislost (pokud nechceme aby některé kolo prokluzovalo po povrchu). Stupeň řiditelnosti je 0, protože žádné z kol nejde natáčet za účelem změny směru pohybu.



Obrázek 4 Hnací síly rychlostí motorů u všesměrových podvozků

Protože stupeň manévrovatelnosti δ_M je 3, potřebujeme k jednoznačnému řízení podvozku tři parametry. V případě těchto všesměrových podvozků jsou to parametry v_x, v_y (vektor rychlosti pohybu) a ω (úhlová rychlost otáčení robotu)



Obrázek 5 Parametry řízení podvozku v_x, v_y a ω

1.2.1 Přímočarý pohyb

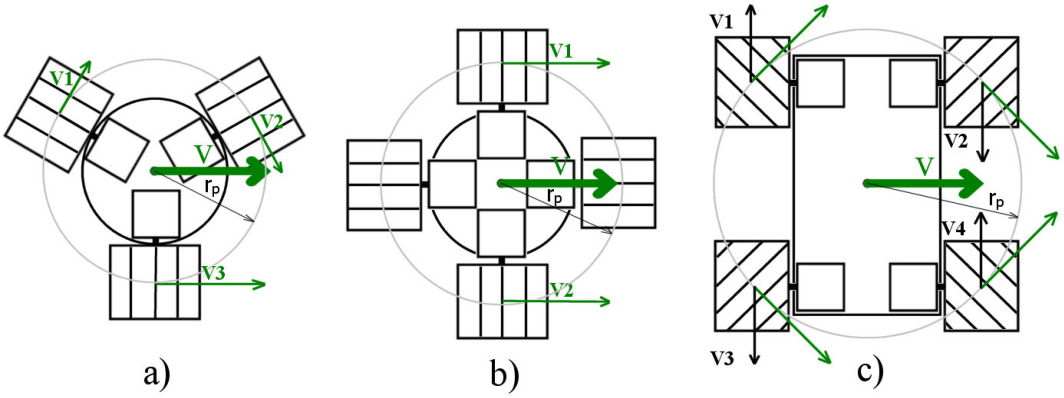
Pro přímočarý pohyb robotu ve směru vektoru rychlosti $\vec{v}$ musíme správně určit obvodové rychlosti kol v_1, v_2, v_3 tak, aby platila podmínka

$$\vec{v} = \sum_{i=1}^{n} \vec{v}_i \tag{1.1}$$

Aby se při přímočarém pohybu zabránilo nežádoucí současné rotaci robota, musí být součet kroutících momentů roven nule. Zjednodušený tvar podmínky za předpokladu, že r_p je shodné pro všechna kola

$$\sum_{i=1}^{n} v_i = 0 \tag{1.2}$$

Stanovit rychlosti v_1, v_2, v_3 tak, aby platily podmínky, není úplně jednoduché, zabývá se tím literatura [2] a [3].



Obrázek 6 Skládání hnacích sil při přímočarém pohybu

Zvláštností podvozku c) je, že se hnací síla vždy vychyluje o 45°, proto je výsledná rychlost dána součtem těchto vychýlených sil, proto i tato varianta podvozku je schopná jezdit i do strany stejně jako ostatní varianty podvozku.

1.2.2 Rotační pohyb

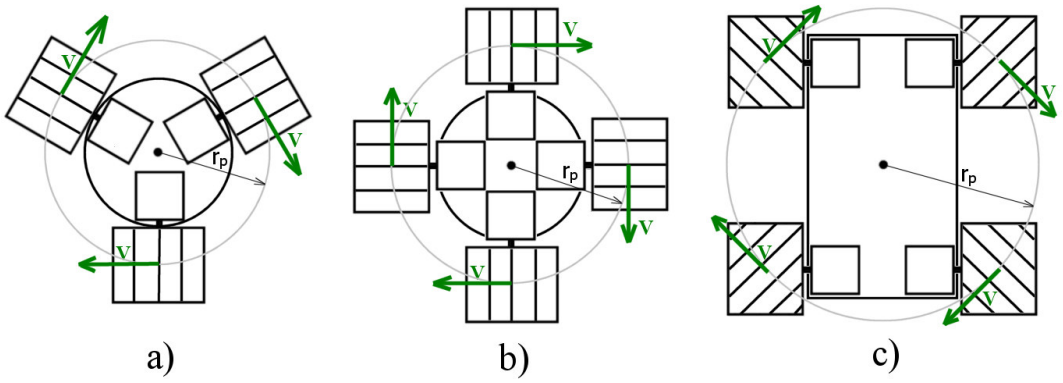
Pro rotační pohyb robota musíme na základě požadované rychlosti otáčení robota ω správně určit obvodové rychlosti kol v_1, v_2, v_3 tak aby platila podmínka

$$\sum_{i=1}^{n} \vec{v}_i = 0 \tag{1.3}$$

To znamená, že vektorový součet rychlostí jednotlivých kol musí být nulový. Nejjednodušším řešením je stanovit stejné obvodové rychlosti pro všechna kola ($v=v_1=v_2=v_3$). Pokud se všechna kola otáčí stejně, otáčí se robot kolem svého středu a můžeme použít jednoduchý vzorec, ve kterém je pouze obvodová rychlost pohybu kol a poloměr usazení kol r_p.

$$\omega = \frac{v}{r_p} \tag{1.4}$$

I v tomto případě mají rozdílné konstrukce stejné manévrovací schopnosti.

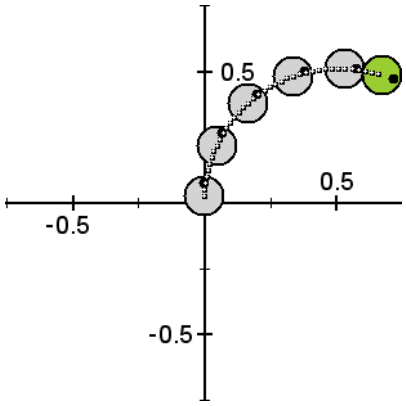


Obrázek 7 Skládání hnacích sil při rotačním pohybu

Pozn. V simulačních schématech v příloze je používán symbol Rp s hodnotou 0.095 m

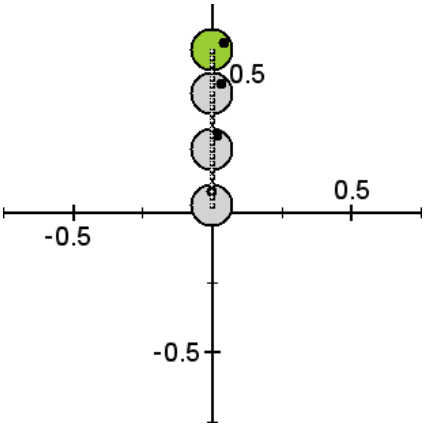
1.2.3 Složený pohyb

Složeného pohybu můžeme dosáhnout tím, že vypočtené hodnoty obvodových rychlostí kol pro přímočarý a rotační pohyb jednoduše sečteme. V tom případě bude mít výsledný pohyb obloukový charakter (podobně jako u automobilu).



Obrázek 8 Složený obloukový pohyb

Druhou variantou složeného pohybu je přímočarý pohyb s nezávislou rotací robota. K dosažení tohoto typu složeného pohybu opět stačí vypočtené hodnoty rychlostí kol jednoduše sečíst. Jediný rozdíl je, že při výpočtu rychlostí kol musíme uvažovat i aktuální hodnotu natočení těla robota a rychlosti kol musíme přepočítávat řádově 20 až 100 krát za sekundu. Tím dosáhneme toho, že trajektorie robota bude složena z velkého počtu velmi krátkých oblouků a ve výsledku bude trajektorie na první pohled vypadat jako přímka.

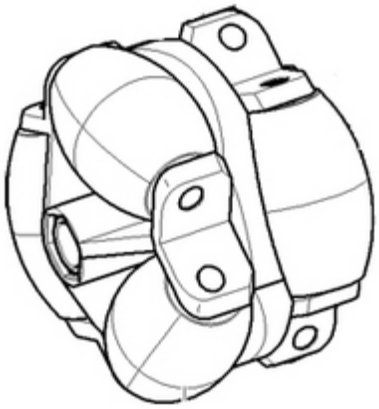


Obrázek 9 Složený přímočarý pohyb s rotací robota

2 KONSTRUKCE ROBOTA

2.1 Typ všesměrových kol

Pro konstrukci robota byla zvolena kola s průměrem 80mm s válečky (resp. soudky) na obvodu o průměru 30 mm kolmo na osu otáčení. Každý soudek je dlouhý 40 mm. Kolo má na obvodu celkem šest soudků ve dvou řadách a tím je zaručeno, že profil kola je přesný kruh (pozn. v simulačních schématech v příloze je poloměr kola reprezentován symbolem Rk s hodnotou 0.04 m)



Obrázek 10 Použitý typ všesměrových kol

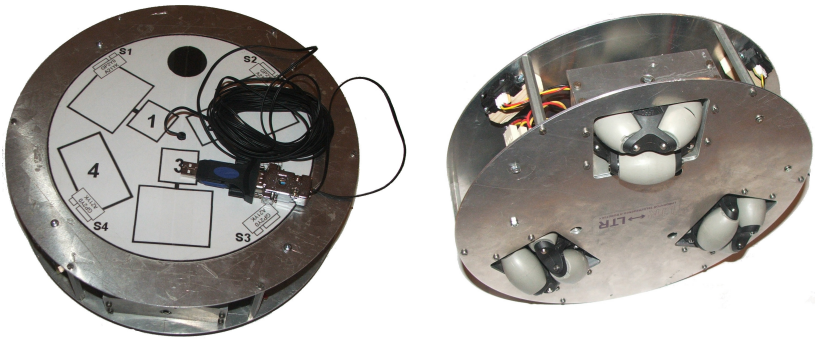
Výběr konkrétního typu všesměrového kola nelze podceňovat, protože výsledná přesnost pohybu robotického podvozku výrazně záleží na tření mezi materiálem podlahy a soudky na obvodu. Zvolený typ všesměrových kol má soudky z tvrdého plastu, který na řadě materiálů prokluzuje (např. na plovoucí podlaze nebo hladkých parketách).

Prokluzování plastových koleček může mít za následek nejen náhodný pokles rychlosti, ale i špatné překonávání velmi nízkých nerovnovností podlahy. V případě prkenné podlahy a výškového rozdíly mezi prkny řádově 1 mm se robot místo šikmého nájezdu na sousední prkno může výrazně stočit do strany- klasická kola s gumovou pneumatikou na obvodu velmi účinně vytváří omezení pohybu do strany, ale všesměrová kola už v principu založena na tom, že klouzání do strany umožňují, proto větší manévrovatelnost zároveň přináší i větší neurčitost pozice robota.

Prakticky tedy bylo zjištěno, že tento konkrétní typ všesměrových kol pro bezchybnou funkci opravdu potřebuje dokonale rovný povrch. Pokud tato podmínka splněna není, je třeba směr korigovat např. manuálně pod dohledem obsluhy nebo pomocí senzorů vyhodnocujících skutečnou polohu robota.

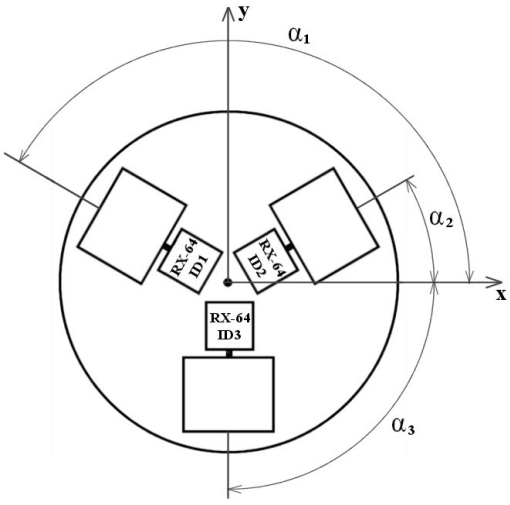
2.2 Mechanické uspořádání robota

Celkem byla použita tři všesměrová kola a tři motory Dynamixel RX-64, každý servomotor ovládá kolo nezávisle na ostatních. Kola mezi sebou svírají úhel 120°. Celkový průměr robota je 30 cm, kola jsou ze tří čtvrtin zapuštěna dovnitř robota, proto jsou ve spodní části pro každé kolo vyříznuty čtvercové otvory 80 x 65 mm (půdorys samotného kola je 80 x 60 mm)



Obrázek 11 Robotický podvozek

Souřadný systém je stejný, jaký se obvykle používá v matematice, tzn. osa X směřuje doprava a osa Y nahoru. Veškeré pohyby robota se zadávají v tomto souřadném systému. Ve stejném souřadném systému jsou definovány i úhlové pozice motorů- motor 1 je umístěn v úhlu 150°, motor 2 v úhlu 30° a motor 3 v úhlu -90°. Stejně tak směr pohybu robota se zadává ve stejném souřadném systému, tzn. pohyb ve směru 0° znamená pohyb doprava a pohyb ve směru 90° znamená pohyb robota směrem nahoru.



Obrázek 12 Použitý souřadný systém

2.3 Použité servomotory

Konstrukce podvozku používá motory Dynamixel RX-64 vyráběné společností Robotis.

2.3.1 Parametry servomotorů Dynamixel RX-64

Motory Dynamixel RX64 jsou servomotory navržené primárně pro stavbu robotů. Někdy se pro servomotory používá i výraz aktuátor, který pochází z anglického slova actuator.

Tento typ servomotoru má v sobě řídící jednotku připojitelnou na sběrnici RS485, proto je možné realizovat připojení až 32 servomotorů s hlavní řídící jednotkou (v našem případě PC) pouze pomocí dvou vodičů, nepovinný třetí vodič pouze propojuje zem.

Obrázek 13 Servomotor Dynamixel RX-64

Každý servomotor obsahuje dva čtyřpinové konektory Molex, takže je možné serva jednoduše řadit do série. Tím se i snižuje množství kabeláže nutné k propojení maximálního počtu 32 motorů.

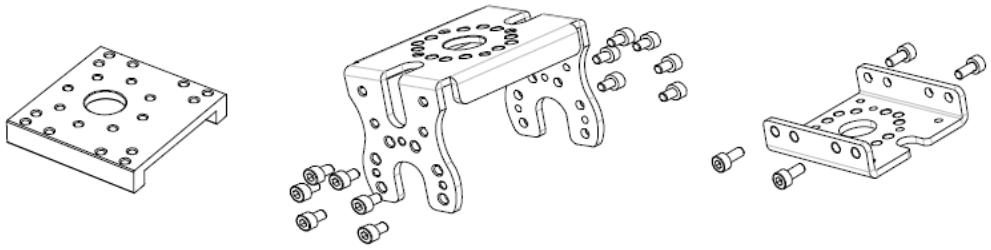
Obrázek 14 Propojení servomotorů do řetězce

Dva piny konektoru Molex využívá sběrnice RS485, další dva piny jsou potřebné k napájení motorů. Ale ani v případě použití převodníku z USB na RS485 není možné napájení z USB portu PC. V technických specifikacích výrobce udává maximální možný proud 1200mA, spotřeba samozřejmě stoupá přímo úměrně hmostnosti, kterou servo nese. V případě našeho konkrétního robota serva nejsou příliš zatěžována, proto se spotřeba jednoho motoru běžně pohybuje kolem 100mA a nikdy nepřesahuje 200mA.

Výrobce servomotorů doporučuje zdroj připojit jako poslední článek řetězce, protože u posledního servomotoru v řetězci vždy zbývá jeden jinak nevyužitý Molex konektor. Kvůli možným proudovým rázům výrobce doporučuje k motorům připojit i kondenzátor 2200μF.

Motory podporují přenosové rychlosti 9600-1000000 Bd, výchozí hodnotou je 57600Bd. Vyšší hodnota zvyšuje chybovost komunikace a nároky na kabeláž, nižší hodnoty zase zbytečně prodlužují dobu potřebnou pro komunikaci, proto byla ponechána výchozí hodnota.

Motory jsou přizpůsobeny ke konstrukci robotů i po mechanické stránce- na plášti motoru jsou malé vystupující lišty s dírami pro snadné připevnění k podkladu nebo dalšímu motoru Dynamixel a to buď přímo nebo přes speciální podložky dodávané výrobcem, tzv. rámečky (v originále „frames“).

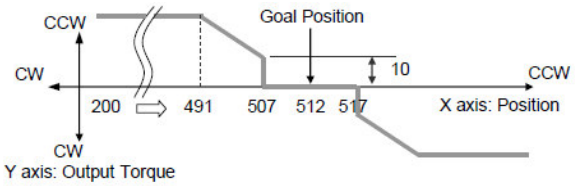


Obrázek 15 Rámečky pro montáž servomotorů

Pomocí všech těchto konstrukčních prvků je možné z těchto motorů sestavit složitější roboty s řadou kloubů, včetně humanoidních robotů.

Obrázek 16 Různí roboti sestavení z motorů Dynamixel RX-64

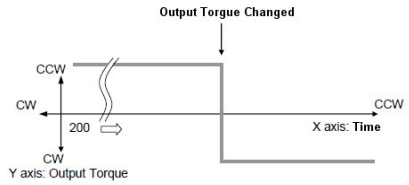
Pro každý kloub je možné do řídící jednotky motoru nastavit bezpečné úhlové meze, proto je takto možné jednoduše zabránit přetěžování motorů nebo poškození robota i v případě chyby v řídícím software. Pro snížení proudových nárazů řídící jednotka plynule zvyšuje kroutící moment (output torque). Na obrázku osa X udává žádanou cílovou polohu (resp.cílový úhel), osa Y udává momentální kroutící moment.



Obrázek 17 Závislost kroutícího momentu na úhlové pozici

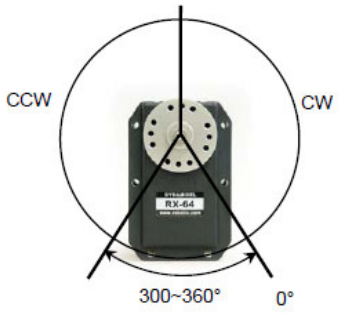
Motor je primárně navržen pro polohování ramen, kde je obvykle vyžadován jen pohyb v určitém rozmezí úhlů. V tomto případě je rychlost i poloha regulována vestavěnou řídící jednotkou.

Motor umožňuje i přepnout i do módu EndlessTurn, tzn. otáčení kolem dokola, které lze použít pro pohon kol. Bohužel v módu EndlessTurn není rychlost nijak regulována, lze nastavovat pouze aktuální kroutící moment bez jakékoli regulace, proto se o plynulé zvyšování a snižování kroutícího momentu musí explicitně starat řídící software na PC.



Obrázek 18 Průběh kroutícího momentu v módu Endless Turn

V módu EndlessTurn (někdy se používá i výraz Wheel mód) se také projevuje fakt, že úhlový senzor polohy pokrývá pouze rozsah 0-300° (senzor poskytuje hodnoty 0 až 1023). Ve slepém úhlu 300-360° jednotka motoru nesprávně vyhodnocuje aktuální pozici a aktuální rychlost- hodnota rychlosti je buď nulová nebo řádově kolem 5000, což je i zcela mimo normální rozsah hodnot (standartní hodnoty rychlosti jsou -1023 až 1023)



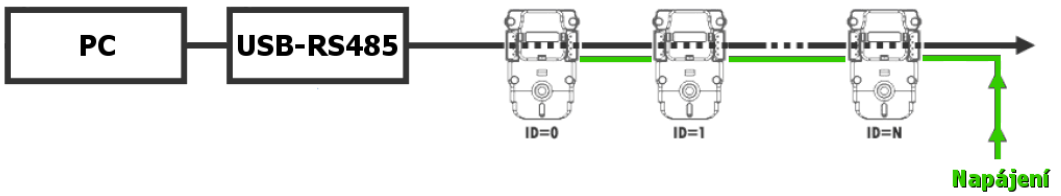
Obrázek 19 Rozsah úhlového senzoru polohy

Tento slepý úhel úhlového senzoru polohy je nejspíš i důvodem, proč se řídící jednotka motoru vůbec nestará o regulaci otáček v EndlessTurn módu. Toto také poněkud ztěžuje regulaci otáček, pokud bychom sami chtěli pokles otáček (např. při vyšší zátěži) dynamicky kompenzovat zvyšováním kroutícího momentu.

V EndlessTurn módu lze kroutící moment nastavovat v rozsahu -1023 až 1023, což přibližně odpovídá úhlovým rychlostem -5.7 až 5.7 rad/s (při zátěži samozřejmě úhlová rychlost klesá).

2.3.2 Komunikace s motory Dynamixel RX64

Jak už bylo zmíněno, všechny motory jsou připojeny k dvouvodičové sběrnici RS485, k PC vedou pouze dva vodiče, zem není v tomto případě nutná, další dva vodiče jsou potřeba k napájení motorů, mezi motory tedy vede celkem čtyřvodičová sběrnice.



Obrázek 20 Motory Dynamixel na sběrnici RS485

K propojení motorů s PC se používá převodník z USB na RS485. Robot byl testován s převodníky Robotis USB2Dynamixel a Premiumcord RS485.

Obrázek 21 Převodník Robotis USB2Dynamixel

Obrázek 22 Převodník Premiumcord RS485

Protože všechny motory jsou připojeny ke sběrnici RS485 defakto paralelně, není možné aby více motorů vysílalo data naráz, proto se důsledně používá komunikace typu master-slave, kde jedna master jednotka (v našem případě software na PC) řídí veškerou komunikaci a ostatní slave jednotky (servomotory) pouze odpovídají na příkazy.

Každý paket vyslaný master jednotkou na sběrnici obsahuje ID motoru, který má paket přijmout. Adresovaný servomotor následně zasílá zpět status paket se žádanými daty. V případě, že je paket určen pro všechny motory (tzv. broadcasting), není status paket vyžadován od žádného z adresovaných servomotorů (protože by nastal konflikt na sběrnici).

Pomocí jednoduchého software „Dynamixel Manager“ dodávaného výrobcem je možné nastavit všechny základní parametry motoru jako baudrate, bezpečný rozsah úhlů a především jeho ID (defaultní je nula). Teprve až po nastavení unikátních ID můžeme všechny motory připojit na jednu společnou sběrnici a začít používat adresování pomocí unikátních ID. Pokud použijeme ID s hodnotou 254, je paket přijat všemi motory.

Komunikace s motory probíhá pomocí paketů, které typicky mají velikost 6 - 12 bajtů, nejvyšší možná velikost je 144 bajtů (to je i velikost vstupního bufferu servomotoru). V následující tabulce je každá buňka jinak široká i když každý parametr zabírá 1 bajt.

Instrukční paket								
255	255	ID	DÉLKA PAKETU	INSTRUKCE	PARAMETR 1	...	PARAMETR N	CRC

Tabulka 1 Tvar instrukčního paketu

Počet parametrů závisí na konkrétním typu instrukce- některá nepotřebuje žádné parametry, u některé může být naopak počet parametrů libovolný. Podporovaných instrukcí je celkem 7.

Instrukce RX64		
Hodnota	Název	Význam
1	PING	Testuje přítomnost motoru na sběrnici
2	READ DATA	Čtení dat (tzn. čtení parametrů)
3	WRITE DATA	Zápis dat (tzn. změna parametrů)
4	REG WRITE	Zápis dat (změna parametrů) synchronizovaná příkazem ACTION
5	ACTION	Sychronizovaná změna už dříve zapsaných parametrů ve všech motorech naráz
6	RESET	Nastavení všech parametrů na výchozí hodnoty
131	SYNC WRITE	Synchronizovaná změna parametrů ve více motorech naráz (bez nutnosti nastavovat každý motor zvlášť a poté volat ACTION)

Tabulka 2 Instrukce podporované servomotory Dynamixel RX-64

Po přijetí paketu řídící jednotka posílá zpět status paket, který obsahuje informaci, jestli se příkaz provedl úspěšně, v případě požadavku na čtení dat jsou ve status paketu obsaženy i parametry 1..N (v závislosti na počtu požadovaných dat). Před vysláním status paketu řídící jednotka motoru čeká po definovanou dobu (Return Delay Time), defaultní hodnota je 0.5ms.

Status paket								
255	255	ID	DÉLKA PAKETU	CHYBA	PARAMETR 1	...	PARAMETR N	CRC

Tabulka 3 Formát status paketu

Parametr CHYBA je nula, pokud se příkaz provedl úspěšně. Pokud je hodnota nenulová, dá se chyba rozkódovat podle následující tabulky.

Chyba			
Bit 7	Nevyužito	Bit 6	Neplatná instrukce
Bit 5	Přetížení motoru	Bit 4	Chyba kontr. Součtu
Bit 3	Neplatný rozsah hodnoty	Bit 2	Přehřátí motoru
Bit 1	Cílový úhel mimo bezp.mez	Bit 0	Napětí mimo dovolený rozsah

Tabulka 4 Význam jednotlivých bitů v parametru CHYBA

Dlouhá řada parametrů má platný rozsah -1023 až 1023, proto jsou na její uložení třeba dvě paměťová místa. 24 z nich je v paměti EEPROM- zde jsou parametry jako BaudRate, ID, ReturnDelayTime, mezní úhly a maximální kroutící moment. V paměti RAM jsou uloženy takové parametry jako aktuální pozice, aktuální rychlost, aktuální zatížení, aktuální teplota.

Obecně tedy většina instrukcí umí pouze číst nebo zapsat libovolný počet dat počínaje definovanou adresou (adresy jednotlivých parametrů najdeme v manuálu) a tím můžeme měnit parametry pohybu servomotoru, např. při zápisu hodnoty 340 (z maxima 1023) do registru nazvaného MovingSpeed se motor roztočí na 33% výkonu a tato nová hodnota je pro řídící jednotku servomotoru platná dokud řídící software nepošle hodnotu jinou.

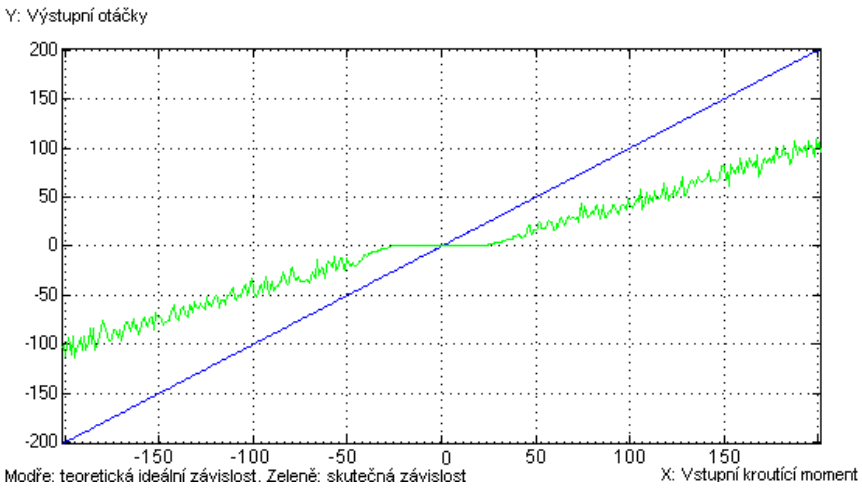
2.3.3 Dynamika motorů Dynamixel RX64

Dynamika motorů nezáleží jen na motoru samotném, ale i na mechanické zátěži, kterou motor musí překonávat, což záleží na konkrétní konstrukci robota. V našem případě motory pohání pouze kola, která mají velmi malou setrvačnost, proto zatížení motorů je malé.

Jako první byla změřena linearita výstupních otáček v závislosti na vstupním kroutícím momentu. Z naměřených hodnot vyplývá, že servomotor má tedy pásmo necitlivosti zhruba mezi kroutícími momenty -25 až 25 a teprve až po překonání vnitřního tření se servomotor začíná roztáčet.

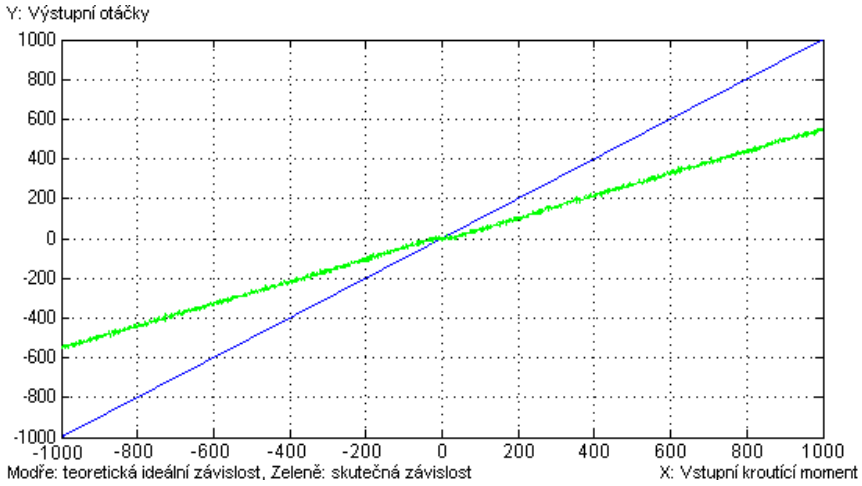
Pásmo necitlivosti -25 až 25 bylo naměřeno pouze u vyjmutého servomotoru. U servomotoru namontovém v robotu bylo naměřeno pásmo necitlivosti větší, od hodnot ±80 až do hodnot ±120. Při prozkoumání mechanické části robota jsem došel k názoru, že by pro lehčí otáčení kol robota prospělo všechna kola odstrojit a na některých částech o něco zvětšit vůle (zvláště na přírubě spojující servomotor a kolo), pravděpodobně by se tím zlepšila přesnost pohybu robota.

Nicméně tato kapitola se zabývá dynamikou motorů Dynamixel jako takových, proto následující grafy nezahrnují tření způsobené zmíněnými nedokonalostmi podvozku.



Obrázek 23 Pásmo necitlivosti u motorů Dynamixel RX-64

Pokud pomineme malé pásmo necitlivosti a zatížení senzoru rychlosti náhodnou chybou, pak se dá závislost výstupních otáček na vstupním kroutícím momentu prohlásit za přibližně lineární.

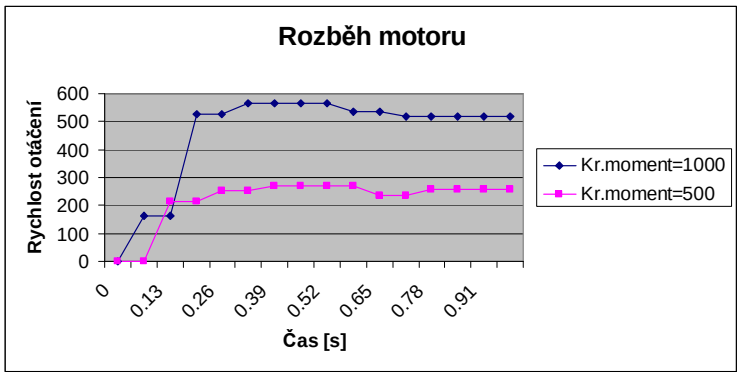


Obrázek 24 Závislost výstupní úhlové rychlosti na vstupním kroutícím momentu

Bohužel ani sám výrobce neudává přesné jednotky- vstupní kroutící moment (osa X) i výstupní úhlová rychlost (osa Y) jsou bezrozměrné jednotky v rozsahu -1023 až 1023.

Dále byla změřena přibližná přechodová charakteristika. Před měřením přechodové charakteristiky byla doba roztočení motoru odhadnuta vizuálním pozorováním zhruba na 0.1 s, proto jako frekvence pro vzorkování dat ze senzoru polohy byla zvolena hodnota 5ms. Při měření hodnot bylo zjištěno, že z nějakého důvodu úhlový senzor polohy mění hodnotu pouze každých 125ms, proto byla vzorkovací frekvence zvýšena na 62,5ms. Následující graf zobrazuje naměřené hodnoty.

Při skokovém přechodu na plný a následně na poloviční výkonu motoru byly naměřeny tyto přechodové charakteristiky.



Obrázek 25 Závislost otáček na čase při skokovém rozběhu motoru

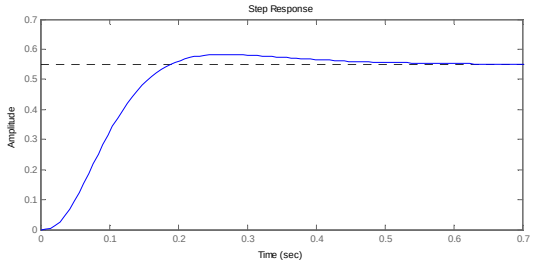
Z grafu je vidět, že se motor roztočí na žádanou hodnotu otáček zhruba za 0.2s, poté rychlost klesne většinou zhruba o 5% a tato hodnota se už pak příliš nemění. Tento pokles není dán ani tak dynamikou pohybu rotoru, nejspíš je to jen chyba soustavy způsobená třením v ložiscích, jehož hodnota neustále kolísá a která se pravděpodobně může u staršího motoru projevovat vice než u nového.

Na základě naměřených dat byla experimentálně stanovena přenosová funkce. Při stanovování přenosové funkce bylo využita znalost základních vlastností přenosů:

- vícenásobné póly ve jmenovateli způsobují pozvolný náběh tvarem připomínající exponenciální funkci, velikost časových konstant ovlivňuje strmost náběhu
- změna hodnot nul v čitateli způsobuje změnu velikosti překmitu

Krátkým experimentováním s hodnotami pólů a nul byla stanovena takováto přenosová funkce servomotoru

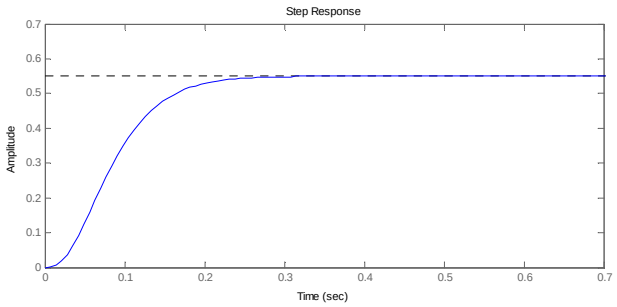
$$F(p) = 0.55 \cdot \frac{0.14p + 1}{(0.1p + 1)(0.04p + 1)^3} \qquad (2.1)$$



Obrázek 26 Přenosová funkce servomotoru včetně poklesu hodnoty

Pokud budeme považovat zmíněný pokles o 5% pouze za malou chybu soustavy, kterou není třeba uvažovat (a kterou bez problémů doreguluje regulátor), vyjde nám jednodušší přenosová funkce

$$F(p) = 0.55 \cdot \frac{1}{(0.03p + 1)^3} \tag{2.2}$$



Obrázek 27 Výsledná přenosová funkce servomotoru

Tento výsledný přenos může reprezentovat dynamické vlastnosti servomotoru v simulačních schématech.

3 SENZORY PRO DETEKCI PŘEKÁŽEK

3.1 Dostupné druhy senzorů

Pro detekci překážek bránících pohybu je žádoucí robota vybavit odpovídajícími senzory. V našem případě je výhodnější koupit již hotové senzory od specializovaných výrobců než vyrábět senzory vzdálenosti z diskrétních součástek. Tyto senzory jsou bez větších problémů dostupné, ceny většiny z nich se pohybují řádově od 300 do 800Kč.

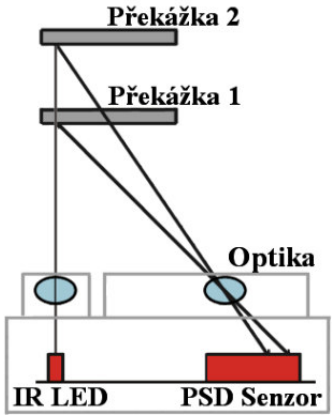
Požadavkům na přesnost měření, dostupnost a cenu vyhovují dvě skupiny senzorů:

a) Optické senzory vzdálenosti
   - měří vzdálenost opticky na principu triangulizace

b) Ultrazvukové senzory vzdálenosti
   - měří vzdálenosti na základě vyslání ultrazvukového pulzu a následném měření času příchodu odraženého signálu

3.1.1 Optické senzory

Optické senzory vzdálenosti fungují na principu triangulizace, závislost přesnosti měření na barvě předmětu podle výrobce existuje, ale je téměř zanedbatelná

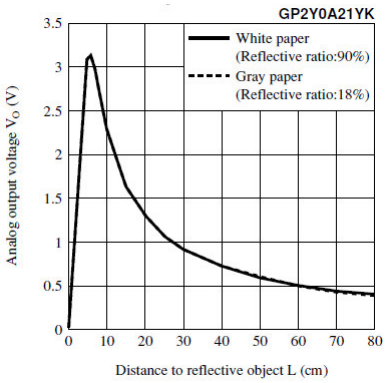


Obrázek 28 Princip optický senzorů vzdálenosti

Optické senzory mají malý vyzařovací úhel (řádově 2°), proto lze přesně určit bod ve kterém se překážka nachází. Z vyzařovacího úhlu vyplývá i omezení nejmenšího rozměru předmětu, který je senzor schopen detekovat- ke správné detekci vzdálenosti musí překážka zastínit celý měřící paprsek, tj. celé 2°

Senzor navenek nemá žádnou inteligentní řídící jednotku, prostě neustále poskytuje aktuální analogovou hodnotu z výstupu PSD senzoru (Position Sensitive Detector), analogový výstup se aktualizuje každých 5 ms (maximální možný čas) bez nutnosti explicitně spouštět měření. Malou nevýhodou je, že výstup je nelineární a vzniká nutnost naměřenou hodnotu napětí softwarově přepočítávat na centimetry.

Výhodou je rychlost měření bez kolize s ostatními senzory stejného typu- pokud se všesměrový robot zatočí, můžeme pomocí dat ze senzorů vykreslit obrys překážky.

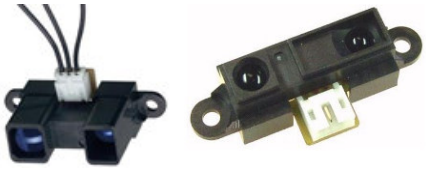


Obrázek 29 Typický průběh analogového výstupu

Kromě senzorů s analogovým výstupem má výrobce i senzory s digitálním výstupem, ale ten pouze spíná z log. 0 na log.1 při pevně nastavené hranici vzdálenosti, kterou není možné měnit (případně ji může přenastavit výrobce). Vyžadované napájecí napětí je 5V, proud u žádného typu nepřekračuje 50mA.

Optické senzory vzdálenosti Sharp		
Typ	Dosah	Výstup
DM2Y0AH01K	4-6 mm	analogový
GP2D120	4-30 cm	analogový
GP2D150A	4-30 cm	digitální (d>15 cm)
GP2Y0A21	10-80 cm	analogový
GP2Y0D21	10-80 cm	digitální (d>24 cm)
GP2YA02YK	20-150 cm	analogový
GP2Y0D02YK	20-150 cm	digitální (d>80cm)
GP2YA700K	100-500 cm	analogový

Tabulka 5 Optické senzory vzdálenosti Sharp



Obrázek 30 Vzhled senzorů vzdálenosti Sharp

3.1.2 Ultrazvukové senzory

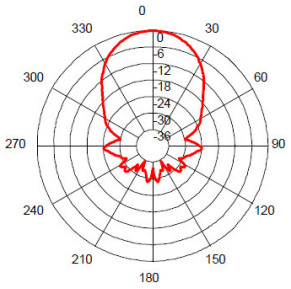
Ultrazvukové senzory měří vzdálenost překážky na základě vyslání ultrazvukového impulsu a následného měření času příchodu odraženého zvukového signálu. Na základě známé rychlosti šíření zvuku se spočítá vzdálenost od překážky.

Výhodou ultrazvukových senzorů obvykle je vyšší měřitelný rozsah a možnost detekce materiálů, které optickým senzorům můžou uniknout (např. sklo nebo jiné průsvitné materiály).

Ultrazvukové senzory vzdálenosti		
Typ	Dosah	Výstup
SRF02 (jednohlavý)	15-600 cm	Sériový nebo I2C
SRF05 (dvouhlavý)	3-400 cm	TTL puls 0.1-25ms
SRF08 (dvouhlavý)	3-600 cm	I2C
SRF10 (dvouhlavý)	3-600 cm	I2C

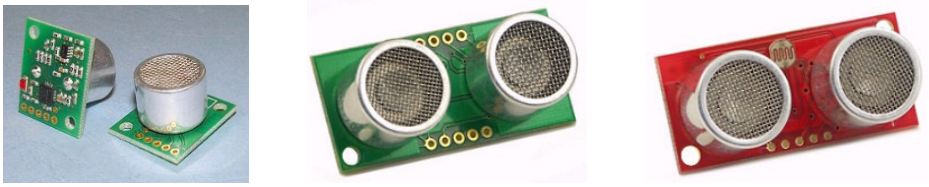
Tabulka 6 Dostupné ultrazvukové senzory

Na rozdíl od optických senzorů mají ultrazvukové senzory vyzařovací úhel řádově 60°, proto nelze přesně určit směr, ze kterého byl odražený signál detekován.



Obrázek 31 Vyzařovací charakteristika ultrazvukového senzoru

Nevýhodou je, že se ultrazvuk od šikmých ploch odráží dále do prostoru (trochu tedy záleží na tvaru překážky), může se odrazit od jiné překážky a vrátit se k úplně jinému senzoru. Proto je doporučeno měřit naráz pouze jedním senzorem a kvůli doznívajícím odrazům mít dostatečnou prodlevu mezi měřeními.

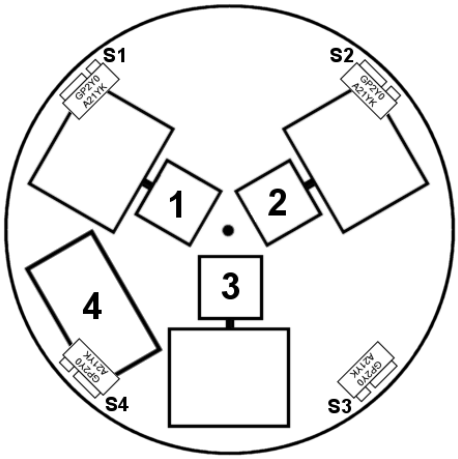


a) SRF 02 **b) SRF05** **c) SRF08**

Obrázek 32 Vzhled ultrazvukových senzorů vzdálenosti Sharp

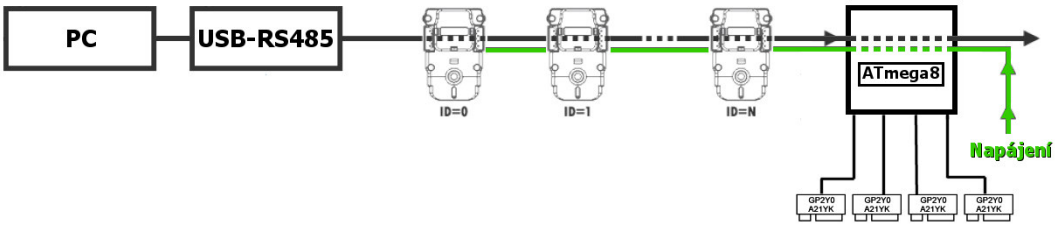
3.2 Připojení senzorů

Na základě srovnání parametrů senzorů byly vybrány senzory Sharp GP2Y0A21 s analogovým výstupem s dosahem 10 až 80 cm. Skutečný dosah senzorů je 6 až 100 cm, tzn. dosah senzorů Sharp je o něco vyšší než výrobcem deklarovaný.



Obrázek 33 Rozmístění senzorů vzdálenosti S1 až S4

Komunikaci analogových senzorů po sběrnici RS485 zajišťuje přídavná elektronická deska s mikrokontrolerem Atmel AVR Mega8 běžícím na frekvenci 11.052MHz (na obrázku čísla 1 až 4 znamenají ID zařízení na sběrnici)



Obrázek 34 Připojení přídavné desky elektroniky na sběrnici RS485

V mikrokontroleru je implementován stejný protokol jako u motorů Dynamixel, proto lze pro čtení dat ze senzorů používat stejnou fyzickou vrstvu i stejný protokol.

Instrukce RX-64		
Hodnota	Název	Význam
1	PING	Testuje přítomnost na sběrnici
2	READ DATA	Čtení dat (tzn. čtení parametrů)
3	WRITE DATA	Zápis dat (tzn. změna parametrů)

Tabulka 7 Instrukce RX-64 podporované senzorickou deskou

U motorů Dynamixel lze nastavovat nebo číst 37 různých parametrů (některé zabírají jeden bajt, některé dva bajty). Senzorická deska nabízí parametrů pouze osm, při pokusu číst jakýkoli jiný parametr se vrátí chyba Instruction error.

Adresa	Původní význam u motoru Dynamixel Nový význam u senzorické desky
24	Přepnutí do EndLessTurn módu
	Zápis hodnoty na port D (konektor MLE14)
25	Rozsvícení/zhasnutí červené LED
	Rozsvícení/zhasnutí červené LED
30	Cílová úhlová pozice (-1023 až 1023)
	Vzdálenost překážky od senzoru 1 v centimetrech (0 až 100)
32	Úhlová rychlost otáčení/Kroutící moment (-1023 až 1023)
	Vzdálenost překážky od senzoru 2 v centimetrech (0 až 100)
34	Horní mez kroutícího momentu
	Vzdálenost překážky od senzoru 3 v centimetrech (0 až 100)
36	Aktuální úhel natočení rotoru (0 až 1023, kde 360° odpovídá hodnotě 1280)
	Vzdálenost překážky od senzoru 4 v centimetrech (0 až 100)
38	Aktuální úhlová rychlost rotoru (0 až přibližně 550)
	Rezervováno pro další analogový senzor
40	Aktuální zatížení motoru (0 až 1023)
	Rezervováno pro další analogový senzor

Tabulka 8 Parametry poskytované senzorickou deskou

Tím, že se senzorická deska navenek chová jako servomotor Dynamixel, se zjednodušuje komunikace na straně PC- software tak může pro komunikaci se servomotory i senzory používat stejný protokol a stejné komunikační knihovny.

3.3 Software v mikrokontroleru

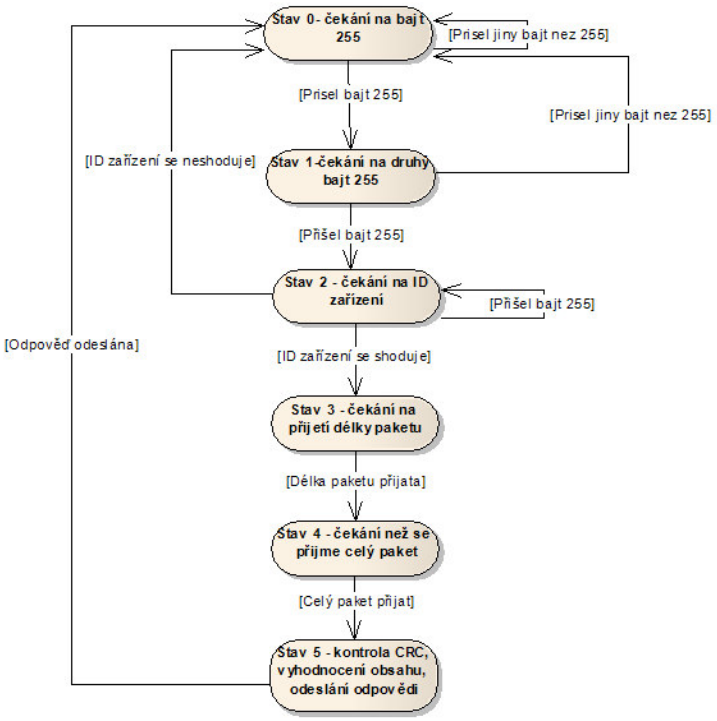
Software v mikrokontroleru je napsán v jazyce C v prostředí AVR Studio. Firmware obsahuje dvě hlavní funkce:

a) skenování hodnot ze senzorů v nekonečné smyčce, přepočet nelineárního výstupu senzoru na lineární hodnotu 0 až 100 cm a následně uložení do vyhrazeného paměťového prostoru, kde jsou parametry uloženy ve stejném pořadí jako v motorech dynamixel.

Napětí [V]	3.2	2.25	1.62	1.1	0.9	0.72	0.59	0.5	0.43	0.4	0.34
Hodnota AD převodníku [1]	1280	900	648	440	360	288	236	200	172	160	136
Vzdálenost [cm]	6	10	15	25	30	40	50	60	70	80	100

Tabulka 9 Linearizační tabulka pro přepočet vzdálenosti na centimetry

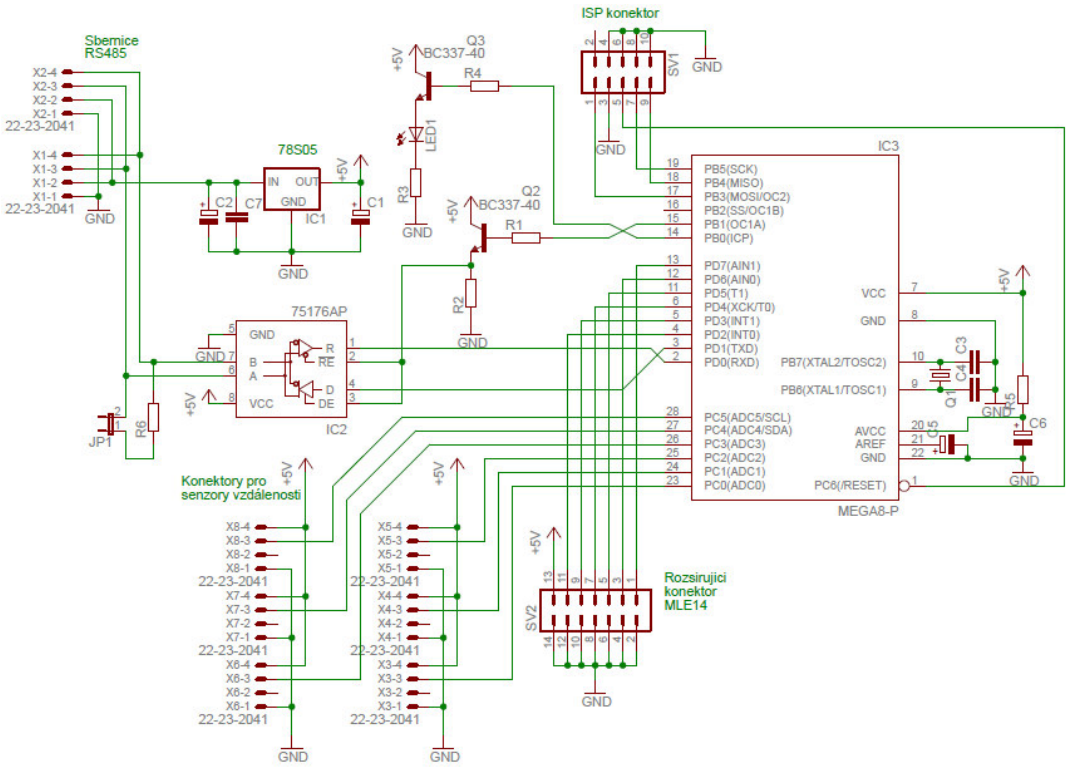
b) při příchodu přerušení z UART kontroleru se vyhodnocuje příchozí bajt pomocí stavového automatu. Poté co automat přijme celý paket včetně správného CRC se zavolá procedura na vyhodnocení obsahu paketu (příkaz PING, READ DATA nebo WRITE DATA) a odešle se příslušná odpověď.



Obrázek 35 Stavový automat pro příjem dat ze sériové linky

3.4 Schéma zapojení senzorické desky

Deska elektroniky obsahuje stabilizátor napětí 5V, protože motory jsou napájeny napětím až 24V, integrovaný obvod 75176 pro konverzi signálů na RS485 (který vyžaduje řízení směru komunikace, proto je využit pin mikrokontroleru PB1), červenou LED, dva čtyřpinové konektory pro připojení k poslednímu motoru Dynamixel, šest čtyřpinových konektrorů pro připojení analogových senzorů (dva nejsou využity a slouží jako rezerva do budoucna), desetipinový konektor MLW10 pro programování mikrokontroleru a čtrnáctipinový konektor, který byl během využit pouze během vývoje pro připojení diagnostických LED.



Obrázek 36 Schéma zapojení senzorické desky

Deska obsahuje i jumper pro zapojení/rozpojení ukončovacího rezistoru 120 ohmů, který podle standartu RS485 má být součástí kabeláže na obou koncích. Ale prakticky bylo zjištěno, že po připojení ukončovacích rezistorů PC přestane přijímat data od servomotorů Dynamixel, zatímco od senzorické desky data přijímá stále.

Proto byly ukončovací odpory na obou koncích kabeláže odpojeny, odpojení ukončovacích rezistorů v případě krátké kabeláže podporuje i literatura [6] a [8].

4 JOYSTICK

Jedním z požadavků na software je ovládání robota joystickem. Joysticky nebo joypady v různých podobách vyrábí velká řada výrobců a každý typ joysticku nebo joypadu je vybaven jiným počtem tlačítek, jiným počtem ovládacích os a v dřívější dobách existovalo i více fyzických rozhraní (Gameport nebo USB), dnes už se používá výhradně USB.

V současné době se považuje za samozřejmé, že každý výrobce joysticku dodává i ovladač pro rozhraní DirectInput (DirectInput je jednou z komponent DirectX). Řídící software pro PC používá pro čtení dat z joysticku rozhraní DirectInput 7 nebo novější a díky tomu tak aplikace může používat libovolný joystick od libovolného výrobce nezávisle na fyzickém rozhraní a parametrech joysticku.

DirectInput 7 definuje pouze maximální počet os, tlačítek, posuvníků a pozičních klouboučků (v originále „points of view“) a je pouze na výrobci, kolik os či tlačítek použijí. Joystick pak systému DirectInput nahlásí kolik os je skutečně připojeno. Celkem jsou podporovány:

-tři poziční osy X, Y, Z (rozsah -1000 až 1000)
-tři rotační osy X, Y, Z (rozsah -1000 až 1000)
-dva posuvníky (rozsah -1000 až 1000)
-čtyři poziční kloboučky (rozsah 0 až 360)
-32 tlačítek (hodnoty true/false)

Dále DirectInput podporuje force-feedback, neboli vibrování- účelem je poskytnout uživateli zpětnou vazbu, například když robot narazí do překážky.

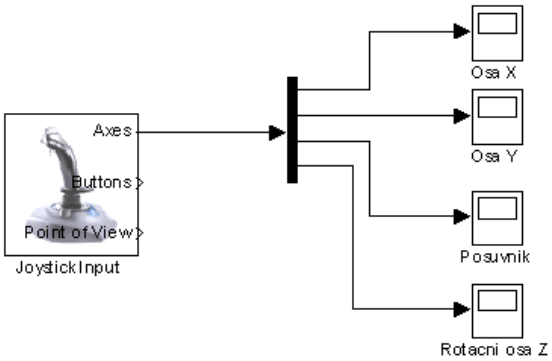
DirectX tedy podporuje celkem osm os (pokud počítáme i posuvníky), v praxi ale žádný výrobce ovladač s tolika osami nenabízí, běžné joysticky mají pouze dvě poziční osy X a Y, lepší joysticky mají navíc posuvníky a třetí rotační osu Z (krut páky zhruba v rozsahu ±15°), zcela běžný je i jeden poziční klobouček. Posuvníky se nejčastěji využívají na pedály brzda a plyn nebo v případě joysticku často vypadají jako malá páčka na spodní části joysticku (na obrázku je to páčka s bílými šipkami), poziční kloboučky se typicky používají například pro přepínání směru pohledu (point of view)..

K ovládání robota byl vybrán joystick se dvěma pozičními osami X Y k ovládání směru jízdy robota (parametry v_x a v_y) a jednou rotační osou Z k ovládání rotace robota (parametr ω)

Obrázek 37 Joystick se třemi osami, jedním posuvníkem,
jedním pozičním kloboučkem (zelený knoflík nahoře) a šesti tlačítky

K přidání podpory joysticku do C# aplikace je třeba získat některou z DLL knihoven pro čtení dat z joysticku a přidat do projektu. Vybraná knihovna je popsána i v kapitole 5.2.4 a se nachází i na přiloženém CD.

K ovládání matematického modelu vytvořeného v prostředí Matlab Simulink joystickem stačí vzít už existující komponentu Joystick z Virtual Reality Toolboxu a použít demultiplexer k získání dat z jednotlivých os, tlačítek a pozičních kloboučků.



Obrázek 38 Joystick v simulačním schématu Matlab Simulink

Na rozdíl od DirectInput všechny osy a posuvníky nabývají hodnot v intervalu $\langle -1,1 \rangle$. Malou nevýhodou je, že komponenta JoystickInput automaticky mění velikost výstupního vektoru podle skutečného počtu os a posuvníků. Tzn. pokud později k již dříve odladěnému simulačnímu schématu připojíme levnější joystick s menším počtem os, zahlásí Matlab chybu, že nelze dva signály multiplexovat na čtyři větve. Proto je vhodné po odladění schématu ve vlastnostech joysticku zrušit volbu „Adjust I/O ports according to joystick capabilities“.

5 ŘÍDÍCÍ SOFTWARE

Řízení všesměrového podvozku znamená především správně stanovit úhlové rychlosti ω_1, ω_2, ω_3 a případně ω_4 na základě vstupních parametrů v_x, v_y a ω. Vstupní parametry jsou načítány:

a) z joysticku, tzn. uživatel joystickem přímo ovládá kam má robot jet
b) z regulátoru polohy, tzn. uživatel předem nadefinuje trasu robota a regulátor polohy dynamicky nastavuje vstupní parametry v_x, v_y a ω.

Jako první krok byl vytvořen matematický model s cílem vytvořit simulační schéma v prostředí Matlab Simulink připojené na reálného robota (tzv. „hardware-in-the-loop“)

5.1 Matematický model v prostředí Matlab

Vzorec pro výpočet rychlostí kol na základě žádaného vektoru rychlost a zpět byl převzat z literatury [2] a [3], pouze byl přizpůsoben souřadný systém. Konstanty γ odpovídají úhlovým pozicím motorů, v našem případě odpovídají hodnotám úhlů 150°, 30° a -90°.

$$\begin{aligned} v_1 &= v \cdot \sin(\gamma_1 - \alpha) + \omega \cdot r \\ v_2 &= v \cdot \sin(\gamma_2 - \alpha) + \omega \cdot r \\ v_3 &= v \cdot \sin(\gamma_3 - \alpha) + \omega \cdot r \end{aligned} \tag{5.1}$$

v - žádaná rychlost přímočarého pohybu
α - žádaný směr přímočarého pohybu
ω - žádaná úhlová rychlost otáčivého pohybu
v_1 - obvodová rychlost prvního kola
v_2 - obvodová rychlost druhého kola
v_3 - obvodová rychlost třetího kola
γ_1 - úhel umístění motoru 1
γ_2 - úhel umístění motoru 2
γ_3 - úhel umístění motoru 3
r - poloměr kružnice, na které jsou kola usazena

Pomocí těchto vzorců lze pro přímočarý pohyb stanovit rozložení hnacích sil do jednotlivých kol.

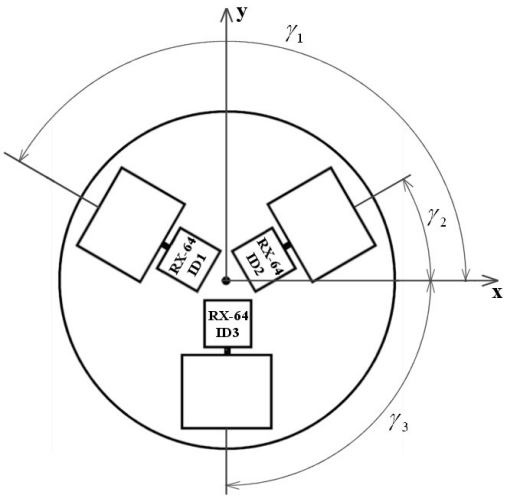
Pomocí druhého vzorce je možné na základě obvodových rychlostí jednotlivých kol určit velikost přímočarého i otáčivého pohybu. Takto je možné integrovat skutečnou polohu robota na základě dat získaných ze senzorů motorů.

$$v_x = \frac{1}{3}(v_1 + v_2) - \frac{2}{3}v_3$$
$$v_y = (v_1 - v2) \cdot \frac{\sqrt{3}}{3} \qquad (5.2)$$
$$\omega = \frac{1}{3}(v_1 + v_2 + v_3)$$

v_x - vypočtená rychlost pohybu v ose X

v_y - vypočtená rychlost pohybu v ose Y

ω - vypočtená úhlová rychlost otáčení těla robota

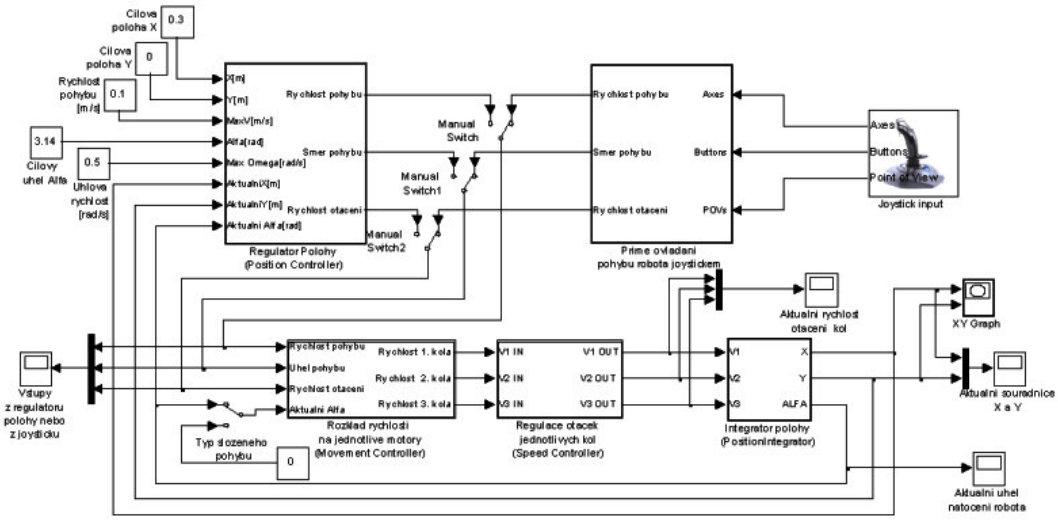


Obrázek 39 Rozmístění os

5.1.1 Simulační schéma v prostředí Matlab Simulink

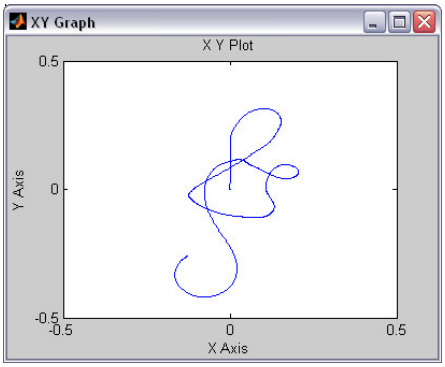
Simulační schéma v Matlabu realizuje všechny zmíněné vzorce včetně integrování rychlostí na polohu robota.

Ve schématu je i několik komponent „Manual switch“, neboli manuálních přepínačů. Tři přepínače nahoře přepínají mezi ovládání z joysticku a polohovým regulátorem. Skutečný software v C# interně obsahuje tyto přepínače také. Za normálních okolností může uživatel ovládat robota joystickem a pouze při spuštění předem naplánované trasy se dočasně přepnou na polohový regulátor. Po dojetí na koncový bod naplánované trasy se přepínače přepnou zpět na ovládání joystickem.



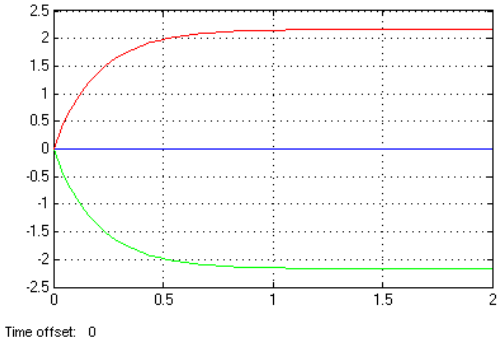
Obrázek 40 Simulační schéma v prostředí Matlab Simulink

Blok „Integrátor Polohy“ na základě rychlostí z jednotlivých motorů integruje aktuální souřadnice robota, což má tu výhodu, že se v Matlabu zobrazuje odhad polohy robota, tím pádem je možné ověřovat správný chod regulátorů i bez fyzicky připojeného robota.

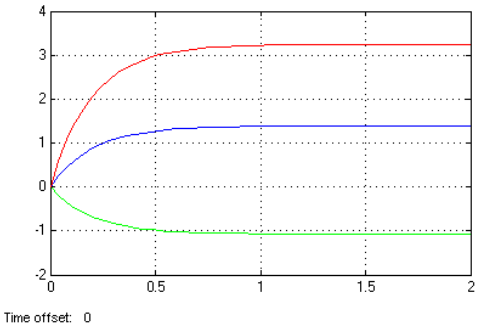


Obrázek 41 Zobrazení výstupu z integrátoru polohy komponentou XY Graph

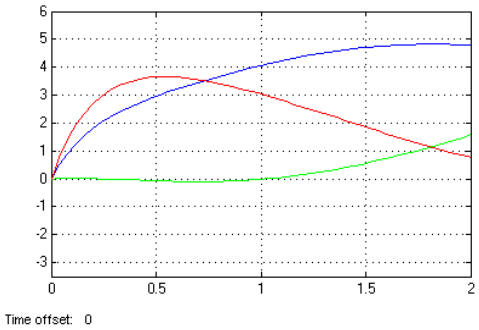
Manuální přepínač „Typ složeného pohybu“ slouží k volbě typu složeného pohybu při ručním ovládání joystickem. Při ovládání polohovým regulátorem tento přepínač musí být přepnutý na vstup z integrátoru polohy. Rozdíl mezi jednotlivými typy složeného pohybu bude nejlépe vidět v následujících grafech.



Obrázek 42 Průběhy úhlových rychlostí při jednoduchém pohybu ve směru osy Y



a) složený obloukový pohyb



b) pohyb přímočarý s nezávislou rotací

Obrázek 43 Průběhy úhlových rychlostí při složeném pohybu

Na obrázku 43 b) je vidět, že regulátor neustále mění hodnoty rychlostí v jednotlivých motorech, aby dosáhl přibližně přímočarého pohybu i přes neustálou rotaci podvozku.

5.1.2 Komunikace s motory

Pro připojení skutečných motorů k simulačnímu schématu potřebujeme knihovnu, která implementuje Dynamixel protokol.

Společnost Robotis vývojářům nabízí knihovny jak pro C# tak i pro prostředí Matlab. Ovladač pro Matlab je nabízen ve formě dll knihovny realizované v jazyce C, ale její použití není jednoduché, navíc není kompatibilní se staršími verzemi Matlabu, proto jsem vyvinul novou knihovnu v jazyce Matlab skript, která se podle mého názoru používá výrazně jednodušeji. Příkazy lze volat i přímo z příkazové řádky Matlabu a lze využívat i starší verze Matlabu. Po registraci adresáře se sadou Matlab skriptů je možné

volat tyto funkce. Jako první je třeba otevřít port příkazem dxl_openport a poté už je možné volat všechny ostatní funkce.

Funkce pro odeslání elementárních příkazů	
dxl_action	Pošle příkaz ACTION
dxl_closeport	Zavírá sériový port
dxl_crc	Počítá CRC před odesláním paketu Dynamixel motoru
dxl_openport	Otevírá sériový port
dxl_ping	Pošle příkaz PING
dxl_readbyte	Pošle příkaz READ a přečte parameter typu bajt
dxl_readword	Pošle příkaz READ a přečte parameter typu word
dxl_regwritebyte	Pošle příkaz REGWRITE a zapíše parametr typu bajt
dxl_regwriteword	Pošle příkaz REGWRITE a zapíše parametr typu word
dxl_reset	Pošle příkaz RESET
dxl_status.m	Použito interně pro čtení odezvy na příkaz
dxl_writebyte	Pošle příkaz REGWRITE a zapíše parametr typu bajt
dxl_writeword	Pošle příkaz REGWRITE a zapíše parametr typu word

Tabulka 10 Matlab příkazy pro odeslání elementárních instrukcí

Teoreticky je možné všechny funkce motorů ovládat pomocí základních elementárních funkcí. Prakticky to ale znamená pamatovat si pro každý parametr číslo registru a jestli je hodnota typu bajt nebo word (např. zápis hodnoty 1 na adresu 25 rozsvítí LED a zápis hodnoty -1023 až 1023 na adresu 32 nastaví úhlovou rychlost). Proto existují tyto pomocné funkce.

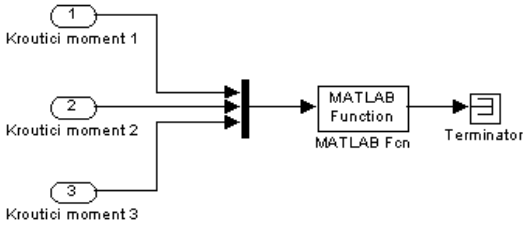
Rozšířené funkce využívající elementární funkce	
dxlex_currentload	Vrací aktuální zátěž motoru (0..1023)
dxlex_currentposition	Vrací aktuální úhlovou pozici (0..1023)
dxlex_currentspeed	Vrací aktuální úhlovou rychlost (-1023 až 1023)
dxlex_distsensors	Vrací hodnoty vzdálenosti ve všech čtyř čidel naráz
dxlex_led	Zapíná/vypíná červenou status LED
dxlex_maxtorgue	Nastavuje maximální možný kroutící moment
dxlex_moving	Vrací 1 pokud je motor v pohybu
dxlex_movingspeed	Nastavuje úhlovou rychlost motoru
dxlex_movingspeeds3	Nastavuje úhlovou rychlost motoru ve všech třech motorech Dynamixel naráz
dxlex_torgueenable	Hodnota 1 přepíná motor do módu Endless turn (mód pro pohon kol)

Tabulka 11 Matlab příkazy pro čtení a zápis různých parametrů

Příklad použití sekvence příkazů v příkazovém řádku Matlabu:

```
port = dxl_openport('COM5');              //otevře port COM5
dxlex_movingspeed(port, 1, 400);          //nastaví motoru 1 rychlost otáčení  400
dxlex_movingspeed(port, 2, 600);          //nastaví motoru 2 rychlost otáčení  600
dxlex_movingspeed(port, 3, -800);         //nastaví motoru 3 rychlost otáčení -800
speed1 = dxlex_currentspeed(port, 1);     //přečte aktuální rychlost motoru 1
dxlex_led(port, 254, 1);                  //rozsvítí LED na všech motorech i na senzorické desce
```

Všechny tyto nové M funkce lze použít i v simulačním schématu vložením bloku Matlab Fcn a přímým zadáním názvu funkce např. „dxlex_movingspeeds3(port, u(1), u(2), u(3))“



Obrázek 44 Komunikace s motory v prostředí Matlab

Všechna simulační schémata používaný tento způsob komunikace se servomotory, proto je nutné před prvním spuštěním simulace vytvořit proměnnou *port* pomocí příkazu `port = dxl_openport(%NAZEVPORTU%);`

Pokud žádný port otevírat nechceme, stačí pouze vytvořit proměnnou `port=0`, která pouze zamezí chybovým hlášením, ale žádný sériový port neotevře.

5.1.3 Nastavení sériového portu

Protože komunikace se servomotory probíhá pomocí velmi krátkých paketů (nejčastěji s délkou 9 bajtů) a tyto pakety se posílají v relativně dlouhých intervalech (cca každých 20 ms), znamená to, že datový tok nabývá velmi nízké hodnoty a v takových případech se operační systém obvykle snaží komunikaci optimalizovat například tím, že čeká na větší množství dat k odeslání a teprve po zaplnění výstupního bufferu data odesílá na fyzické médium.

Ve výsledku pak servomotory můžou přijímat datové pakety v nepravidelných intervalech a chování motorů ve spojení se simulačním schématem v prostředí Matlab Simulink se pak může stát špatně předvídatelné.

Proto je třeba po nainstalování ovladačů převodníku z USB na RS485 nastavit parametry portu na hodnoty doporučené výrobcem servomotorů. Konkrétně výrobce doporučuje snížit velikosti vstupního i výstupního bufferu na hodnotu 4kB nebo nižší. Motory Dynamixel podporují maximální velikost paketu 144 bajtů, proto se jako dostatečná velikost bufferu jeví hodnota 256 bajtů. Výrobce stejně tak doporučuje snížit hodnotu Latency timer na hodnotu 1ms. Po nastavení těchto hodnot by měla být zaručena komunikace v reálném čase.

5.2 Software pro PC

Uživatelské rozhraní pro PC bylo napsáno v jazyce C#. Na software byly kladeny požadavky:

- uživatelsky příjemné rozhraní
- možnost přímého ovládání robota joystickem
- možnost naplánovat trasu robota, tzn. vytvořit polohový regulátor
- zobrazení trajektorie robota nezávisle na tom, jestli je robot ovládán joystickem nebo polohovým regulátorem
- vizualizace dat poskytovaných servomotory (napětí, teplota, zátěž atd.)
- vizualizace dat poskytovaných senzory vzdálenosti
- možnost částečné funkčnosti software i bez připojeného robota
- možnost ovládání robota i bez připojeného joysticku

5.2.1 Struktura software

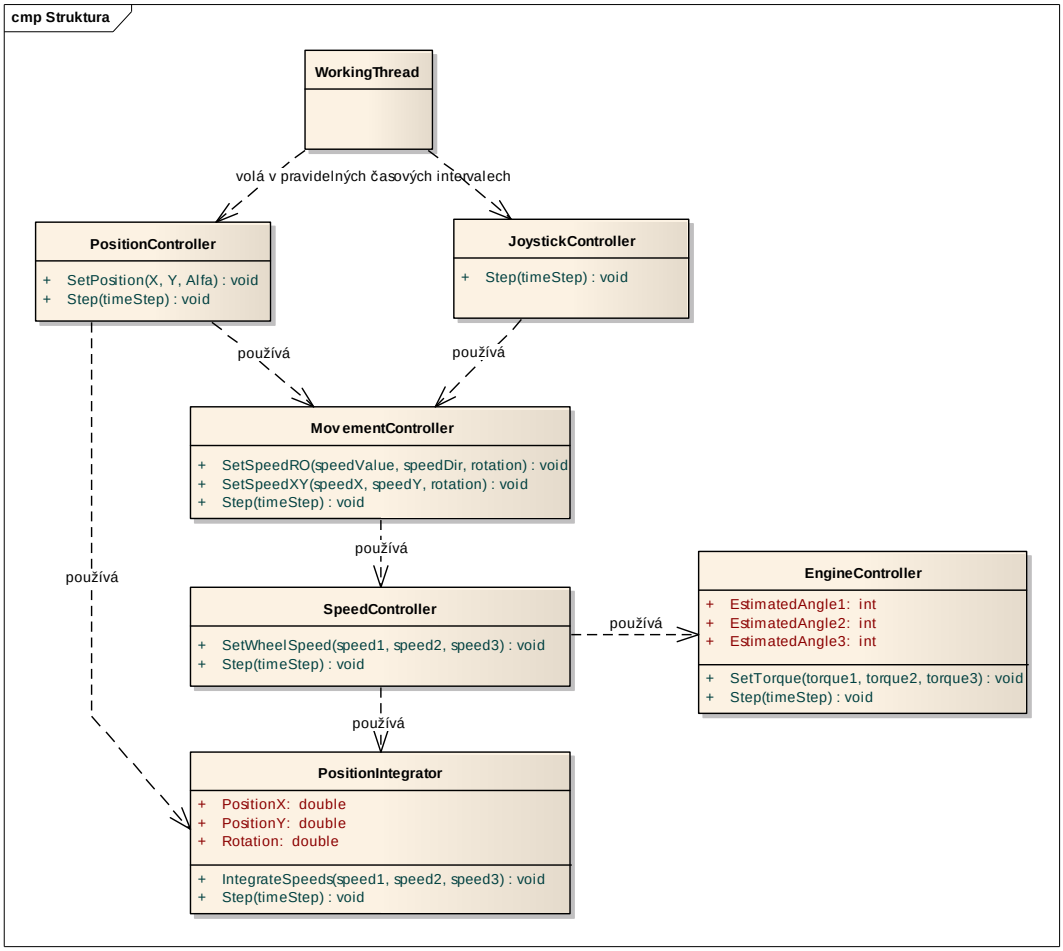
Matematický model implementovaný v jazyce C# používá diskrétní přenosové funkce, u kterých je správná funkčnost podmíněna tím, že se výpočet provádí v přesně daných časových intervalech, proto software běží ve dvou vláknech:

- vlákno s vyšší prioritou se stará o veškerou komunikaci s robotem, senzory a joystickem. Na data získaná při komunikaci se použije matematický model a stanoví se nové úhlové rychlosti motorů, data se zároveň uloží, aby je mohlo používat i druhé vlákno zobrazující data v GUI. Komunikace probíhá tak, aby každý komunikační cyklus trval stejnou dobu a tím pádem aby vzorkovací perioda v matematickém modelu byla co nejvíce konstantní

- vlákno s normální prioritou se stará o zobrazení GUI, obsluhu událostí myši a klávesnice. Vlákno nesmí s motory Dynamixel komunikovat přímo (docházelo by k dočasnému blokování komunikace v prvním vlákně), vlákno vždy využívá už získaná data od prvního vlákna.

Při psaní software byl kladen velký důraz i na modularitu, proto jsou třídy rozděleny do stejných bloků jako simulační schéma. Díky tomu lze snadno najít, kde je blok simulačního diagramu implementován v aplikaci pro PC.

První vlákno s vyšší prioritou pravidelně volá **PositionController** pokud zrovna běží předplánovaná trasa nebo **JoystickController**, pokud zrovna uživatel robota ovládá ručně. Do **MovementControlleru** vstupuje žádaný vektor rychlosti a úhlová rychlost rotace těla robota a výstupem jsou úhlové rychlosti jednotlivých kol.



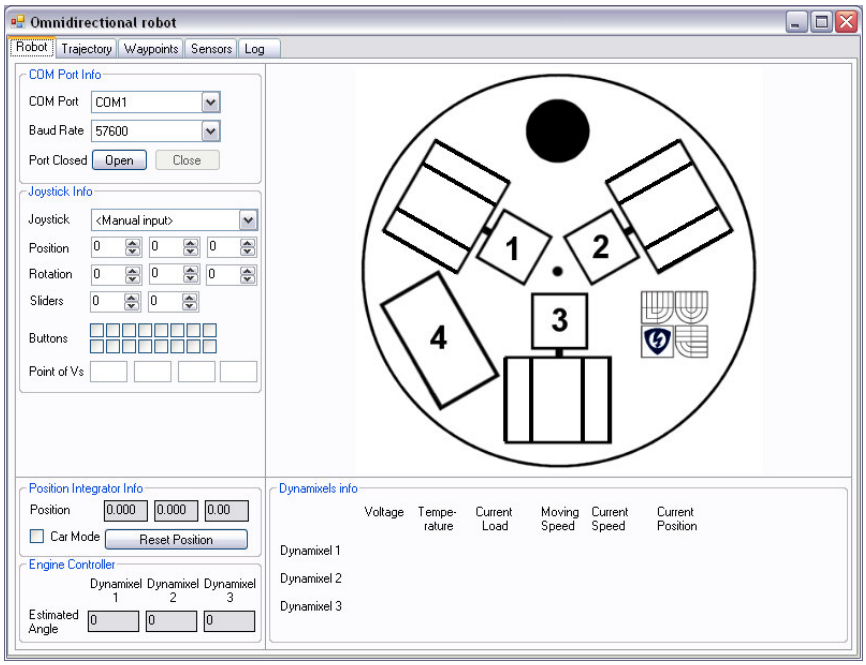
Obrázek 45 Struktura tříd

SpeedController obsahuje diskrétní zpožďovací článek pro plynulejší rozběh a doběh motoru, případně zde může být další dodatečná regulace. **EngineController** na základě požadované úhlové rychlosti nastavuje kroutící moment a zároveň v případě nepřipojeného robota na základě kroutících momentů a délky časových kroků timeStep odhaduje aktuální úhel natočení rotoru motoru. Na základě těchto odhadů úhlů GUI může zobrazovat vizualizaci kol i v případě že data z úhlových senzorů motorů nejsou dostupná. **PositionIntegrator** integruje žádané úhlové rychlosti kol na pozici a natočení robota.

JoystickController kromě čtení dat z joysticku navíc ještě kontroluje data ze senzorů vzdálenosti a pokud je překážka blíž než 15cm, pak se robot zastaví a je možné se od překážky pouze vzdálit.

5.2.2 Uživatelské rozhraní

Uživatelské rozhraní je rozčleněno do pěti záložek



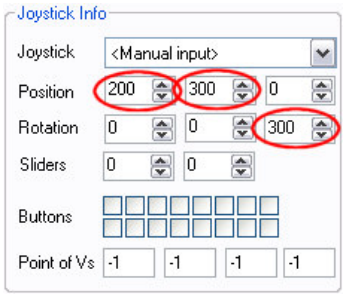
Obrázek 46 Vzhled uživatelského rozhraní

5.2.2.1 Záložka Robot

Záložka Robot umožnuje zvolit port pro komunikaci s robotem a joystick pro ovládání robota, případně je možné zadávat směr číselně pomocí klávesnice a myši. Dále zobrazuje všechny dostupné informace o stavu robota včetně animace všech kol.

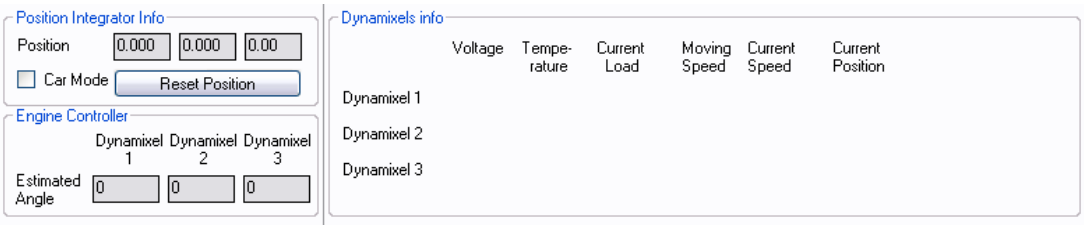
Skupina COM Port Info umožňuje nastavit parametry připojení k robotu, typicky stačí vybrat pouze správný port a stiknout Open.

Skupina JoystickInfo umožňuje robota ovládat jednak ručně přímým číselným zadáním hodnot směrů a jednak pomocí joysticku. Jsou zde zobrazeny stavy všech os včetně těch, které ovladač ani aplikace nepodporuje. V případě, že je v comboboxu vybrán joystick, jsou zobrazené hodnoty pouze pro čtení. V případě, že je zvolen „Manual Input“, je možné stavy os a tlačítek měnit ručně.



Obrázek 47 Ruční zadání směru pohybu a rotace

Ve zbývajících skupinách ve spodní části se zobrazují různé informativní údaje přicházející z motorů Dynamixel, které nelze editovat.

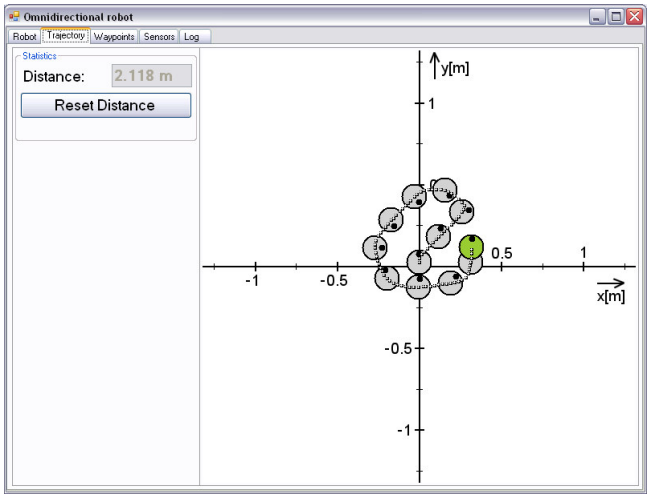


Obrázek 48 Zobrazení stavu servomotorů ve spodní části

Všechna data ve spodní části jsou pouze pro čtení. Car Mode checkbox volí typ složeného pohybu (viz obrázky 5.5 a) a 5.5 b)

5.2.2.2 Záložka Trajectory

Po otevření záložky Trajectory se začne zaznamenávat pozice robota z integrátoru polohy a na základě těchto dat se začne vykreslovat ujetá trajektorie. Díky tomuto principu nezáleží, jestli robota ovládá uživatel joystickem, zadává směr pohybu číselně nebo jestli robotem hýbe regulátor polohy.



Obrázek 49 Zobrazená trajektorie robota

5.2.2.3 Záložka Waypoints

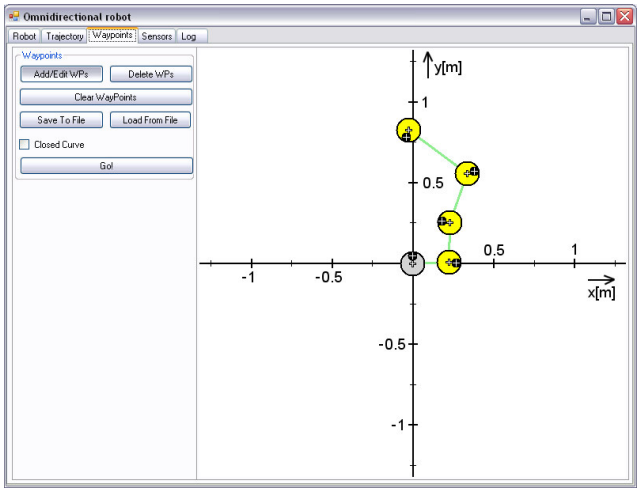
Záložka waypoints umožňuje plánovat trasu robota. Úprava trajektorie funguje ve dvou módech:

- přidávání a úprava waypointů (tlačítko „Add/Edit WPs“ musí být zamáčknuté)

  - Levé tlačítko přidává waypointy, pravým tlačítkem lze body posouvat, případně ikonku robota lze pravým tlačítkem i otáčet.

- mazání waypointů (tlačítko Delete WPs musí být zamáčknuté)

  -Levé tlačítko maže waypointy

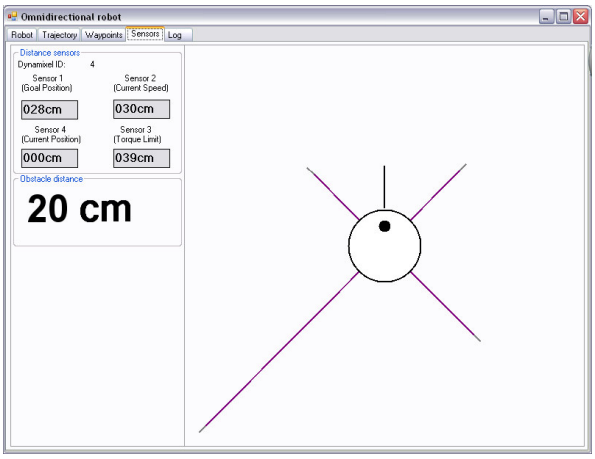


Obrázek 50 Naplánovaná trajektorie robota

Po stisku tlačítka Go! se začne robot pohybovat a zároveň se robot začne na obrazovce animovat (pozn. poloha robota se zobrazuje podle integrátoru polohy skutečného robota, proto se při chybě v regulátoru polohy bude na obrazovce vykreslovat reálný pohyb robota a nikoli pohyb naplánovaný)

5.2.2.4 Záložka Sensors

Záložka Sensors zobrazuje stavy optických senzorů vzdálenosti jednak číselně a jednak graficky. Měřící paprsky jsou zobrazeny fialově, odhad vzdálenosti robota od překážky (interpolovaný z nejbližších senzorů) se zobrazuje jako černý paprsek, ale pouze při pohybu, protože pokud robot stojí, nevíme ve kterém směru a z kterých senzorů se má vzdálenost robota od překážky interpolovat.



Obrázek 51 Vykreslení měřících paprsků a výsledného odhadu vzdálenosti

5.2.2.5 Záložka Log

Záložka Log zobrazuje chybová hlášení pokud se nějaká akce nepovede, např. selhání komunikace, selhání otevření portu atd.

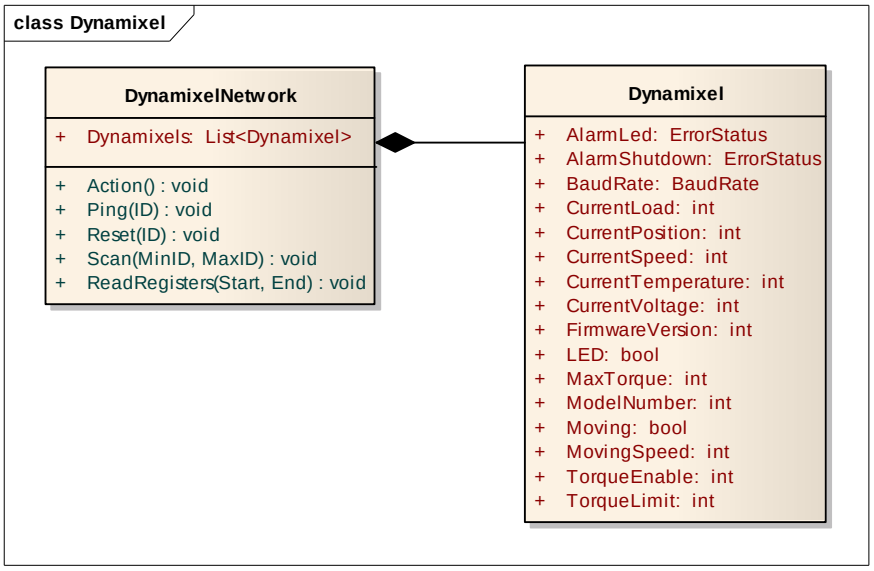
5.2.3 Knihovny pro komunikaci

Software komunikuje se servomotory Dynamixel přes přídavnou DLL knihovnu pro jazyk C#, která implementuje Dynamixel protokol. Tuto knihovnu lze najít na stránkách výrobce Robotis. Tím, že je knihovna implementována přímo v jazyce C#, je velmi jednoduché ji přidat do projektu a i používání je velmi pohodlné.

Knihovna obsahuje sadu třinácti tříd, které obsahují třídy pro podporu dvou hlavních funkcí:

- třídy pro komunikaci se servomotory
- třídy a struktury pro popsání kloubových spojení mezi rameny
- třídy pro polohování robotických ramen, včetně přehrávání předem připravené sekvence pohybů

Protože servomotory používáme v módu EndlessTurn (neboli mód pro pohon kol), nebudou nás zajímat třídy a struktury určené k polohování robotických ramen. Pokud takové třídy vynecháme, zbydou pouze dvě třídy Dynamixel a DynamixelNetwork.



Obrázek 52 Zjednodušený třídní diagram knihovny pro C#

Koncový uživatel se díky této knihovně nemusí starat o dění na pozadí, stačí pouze kterémukoli servomotoru změnit jakýkoli parametr a nadřízená třída DynamixelNetwork se už sama postará o komunikaci a odeslání nové hodnoty servomotoru, pouze v případě registrovaného zápisu (což v podstatě znamená sychronizovaná změna hodnot ve všech servomotorech naráz) musí navíc zavolat ještě příkaz Action. Tyto knihovny tedy umožňují velmi rychle zakomponovat komunikaci se servomotory do vlastní aplikace a není nutné ladit detaily komunikačního protokolu,

Ale taková komunikace je konkrétně v našem případě velmi neefektivní – v našem případě je regulace realizována v software běžícím na PC a v tom případě aby matematický model fungoval správně, je třeba udržovat mezi PC a servomotory konstantní průtok dat a konstantní vzorkovací periodu.

V úplně první verzi software bylo zobrazování stavu servomotorů řešeno pomocí časovače s periodou 0.5 s a obsluha událostí časovače byla řešena přibližně takto:

```
public void updateInfoTimer_Tick(object sender, EventArgs e)
{
     textEdit1.Text = DynamixelNetwork.Dynamixels[0].Voltage;
     textEdit2.Text = DynamixelNetwork.Dynamixels[0].Temperature;
     textEdit3.Text = DynamixelNetwork.Dynamixels[0].CurrentLoad;
     textEdit4.Text = DynamixelNetwork.Dynamixels[0].CurrentSpeed;
     textEdit5.Text = DynamixelNetwork.Dynamixels[0].CurrentPosition;
     textEdit6.Text = DynamixelNetwork.Dynamixels[1].Voltage;
     …
     textEdit15.Text = DynamixelNetwork.Dynamixels[2].CurrentPosition;
}
```

Celkem zde tedy je 15 příkazů pro aktualizaci hodnot zobrazených v GUI. Slabinou ale je, při každém čtení jakéhokoli parametru se servomotoru posílá příkaz READ DATA a čeká se na odpověď po dobu 10 až 15ms, tzn. aktualizace 15 parametrů v GUI každých 500 ms způsobí prodlevu v komunikaci řádově 150 až 225 ms.

Proto byla veškerá komunikace se servomotory výrazně předělána, aby se zajistil co nejplynulejší tok dat. Klíčem jsou metody SyncWrite a ReadRegisters třídy DynamixelManager a změna pořadí čtení parametrů. To přináší výhody:

a) metoda SyncWrite umožňuje nastavit různé rychlosti otáčení u všech tří motorů naráz pomocí broadcastingu (proto se ani nemusí čekat 3x15ms)

b) metoda ReadRegisters umožňuje přečíst více parametrů naráz, proto je možné ze servomotoru načíst naráz jedním příkazem všechny parametry, které nás zajímají, čtení více parametrů naráz trvá téměř stejnou dobu jako čtení pouze jednoho parametru.

Po optimalizaci komunikace je tedy komunikační sekvence takováto:

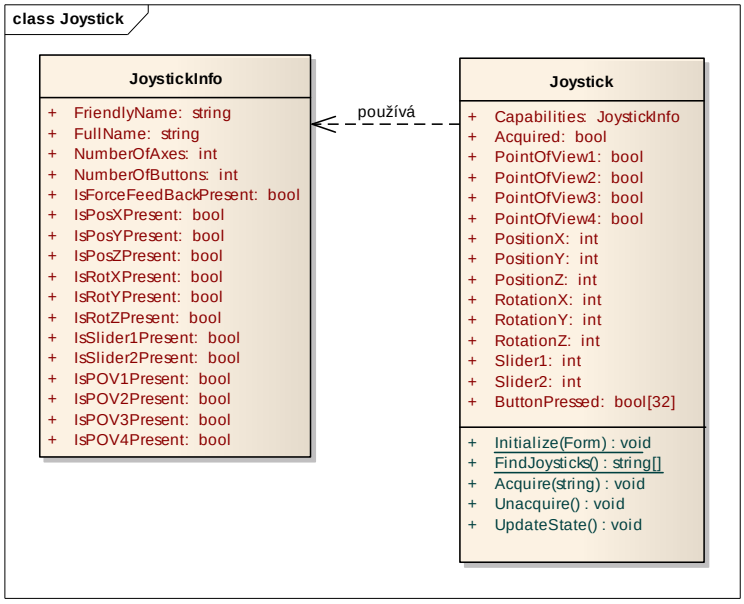
1. načtení směru z joysticku, výpočet rychlostí, odeslání úhl. rychlostí motorům
2. čtení stavu servomotoru č.1
3. načtení směru z joysticku, výpočet rychlostí, odeslání úhl. rychlostí motorům
4. čtení stavu servomotoru č.2
5. načtení směru z joysticku, výpočet rychlostí, odeslání úhl. rychlostí motorům
6. čtení stavu servomotoru č.3
7. načtení směru z joysticku, výpočet rychlostí, odeslání úhl. rychlostí motorům
8. čtení stavu senzorů ze senzorické desky (jakoby servomotor č.4)

Po takovéto optimalizaci komunikace přestala rychlost kolísat a perioda komunikace se ustálila zhruba na čase 0.02 s s tolerancí několik ms, proto je nyní možné ve všech použitých diskrétních přenosových funkcí nastavit vzorokovací periodu na 0.02 s .

5.2.4 Čtení dat z joysticku

.NET Framework sám o sobě neobsahuje žádné knihovny pro čtení dat z joysticku (na rozdíl od Matlabu, který podporu joysticků obsahuje ve Virtual Reality Toolboxu), proto je nutné to řešit přidáním nějaké knihovny od třetí strany.

Nakonec byla použita knihovna Joystick.dll, kterou jsem vyvinul před šesti lety já sám v jazyce C++. Knihovna používá DirectInput verzi 7 (tzn. součást DirectX7). Na rozdíl od jiných knihoven má tato knihovna výhodu, že funguje i s čistou instalací Windows XP, kde není nainstalována žádná aktualizovaná verze DirectX. Fakt, že je knihovna vyvinutá v C++ je kompenzován obalovací třídou napsanou v jazyce C#, což velmi výrazně usnadňuje použití této knihovny.



Obrázek 53 Obalovací třída v C# pro knihovnu Joystick.dll napsanou v C++

ZÁVĚR

Hlavním cílem této semestrální práce bylo vytvoření software pro už existující model robotického podvozku počínaje stanovením matematického modelu a konče implementací software v jazyce C#.

Jako první jsem začal hledáním odborné literatury a na jejím základě jsem v prostředí Matlab Simulink vytvořil simulační schéma s polohovým regulátorem a matematickým modelem pro tento typ podvozku.

Dále jsem od výrobce Robotis stáhnul komunikační knihovnou pro motory Dynamixel pro prostředí Matlab implementovanou v jazyce C. Tuto knihovnu jsem ohodnotil jako problematickou, nekompatibilní se staršími verzemi Matlabu a celkově nevyhovující. Proto jsem implementoval zcela novou komunikační knihovnu pro motory Dynamixel v jazyce Matlab skript. Pro načítání dat z joysticku v prostředí Matlab jsem použil Virtual Reality Toolbox.

Dále jsem rozšířil hardware robota o senzory pro detekci překážek. K připojení senzorů na sběrnici jsem vyvinul elektronickou desku na bázi mikrokontroleru Atmel AVR Mega8, vyvinul firmware v jazyce C a nainstaloval desku i senzory do robota.

Nakonec jsem vytvořil uživatelsky přívětivý software pro demonstrování možností podvozku. Pro komunikaci s motory Dynamixel jsem použil dll knihovnu od výrobce Robotis implementovanou v jazyce C#. Pro načítání dat z joysticku jsem použil knihovnu Joystick.dll, kterou se já sám implementoval v jazyce C++ před šesti lety.

Během práce na robotu jsem zjistil, že otáčky motorů často kolísají řádově až o pět procent, ale co je horší, díky vnitřním třením se někdy motory neroztočí ani při žádosti roztočit se na 15% výkonu. Proto by podle mého názoru prospělo v mechanických částech zvětšit vůle a dále by možná prospělo zavést zpětnovazební regulaci otáček Kvůli poměrně velké neurčitosti pozice robota (kvůli prokluzům po podlaze) se jako vhodné téma pro pokračování jeví sebelokalizace robota.

Z časových důvodů má řídící software projíždění naplánované trasy implementováno velmi jednoduše, proto robot v místě každého waypointu brzdí na nulovou rychlost a znovu se rozjíždí. Proto jako další možné téma se jeví průjezd dané trasy pokud možno konstatní rychlostí s možností vyhýbat se překážkám.

Literatura

[1] Raul Rojas: *A short history of omnidirectional wheels*, 4 s. Dostupné na URL http://robocup.mi.fu-berlin.de/buch/shortomni.pdf

[2] Heldér P. Oliveira, Armand J.Sousa, A. Paulo Moreira, Paulo J. Costa: *Modeling and Assessing of Omni-directional Robots with Three and Four Wheels*, Universidade do Porto, Faculdade de Enhengaria, Portugal, 24 stran, Dostupné na URL http://repositorio-aberto.up.pt/bitstream/10216/20859/2/21325.pdf

[3] F. Ribeiro, I. Mountinho, P.Silva, C.Fraga, N.Pereira: *Three omnidirectional wheels control on a mobile robot*, Groupe de Automacao e Robotica, Departamento de Electronica Industrial, Universidade do Minho, Campus de Azurem, 4800 Guimaraes, Portugal, 5 stran. Dostupné na http://repositorium.sdum.uminho.pt/bitstream/1822/3293/1/46%2520paper%2520for%2520control%25202004%2520-%2520formatado.pdf

[4] Robotis Co. Ltd: *Dynamixel RX-64 User's manual v1.10*, Robotis Co. Ltd., 49 stran, Na internetu už není dostupné, uloženo na CD přílohu

[5] Robotis Co. Ltd: *USB2Dynamixel User's manual v1.2*, Robotis Co. Ltd., 23 stran. Na internetu už není dostupné, uloženo na CD přílohu

[6] Redakce HW.CZ: *RS 485 & 422*,[online] Leden 1998, Dostupné na URL http://www.hw.cz/teorie-a-praxe/dokumentace/rs-485-422.html

[7] Lammert Bies: *RS485 serial information*[online], Dostupné na URL: http://www.lammertbies.nl/comm/info/RS-485.html#topo

[8] Ing. Pavel Poucha: Přenos dat po linkách RS485 a RS422 [online], Dostupné na URL http://www.rs485.cz/papouch1.htm

[9] JUHÁS, M. *Senzorový systém pro mobilní robot.* Brno: Vysoké učení technické, Fakulta elektrotechniky a komunikačních technologií, 2009. 51 stran, Vedoucí bakalářské práce Ing. Zbyněk Fedra, Ph.D.

[10] Microsoft Corporation: *DevCenter – DirectInput* [online] Dostupné na URL: http://msdn.microsoft.com/en-us/library/windows/desktop/ee416842(v=vs.85).aspx

[11] Technická dokumentace Atmel AVR Mega8, 307 stran, dostupné na URL: http://www.gme.cz/dokumentace/432/432-027/dsh.432-027.1.pdf

[12] Technická dokumentace SN75176, 12 stran, dostupné z URL: http://www.gme.cz/dokumentace/433/433-014/dsh.433-014.1.pdf

[13] Technická dokumentace senzoru Sharp GP2Y0A21YK, 4 strany, dostupné na URL http://www.gme.cz/dokumentace/539/539-014/dsh.539-014.1.pdf

[14] Roland Siegwart, Illah R. Nourbakhsh, Davide Scaramuzza: *Introduction to Autonomous Mobile Robots*, ISBN 0-262-01535-8

Seznam příloh

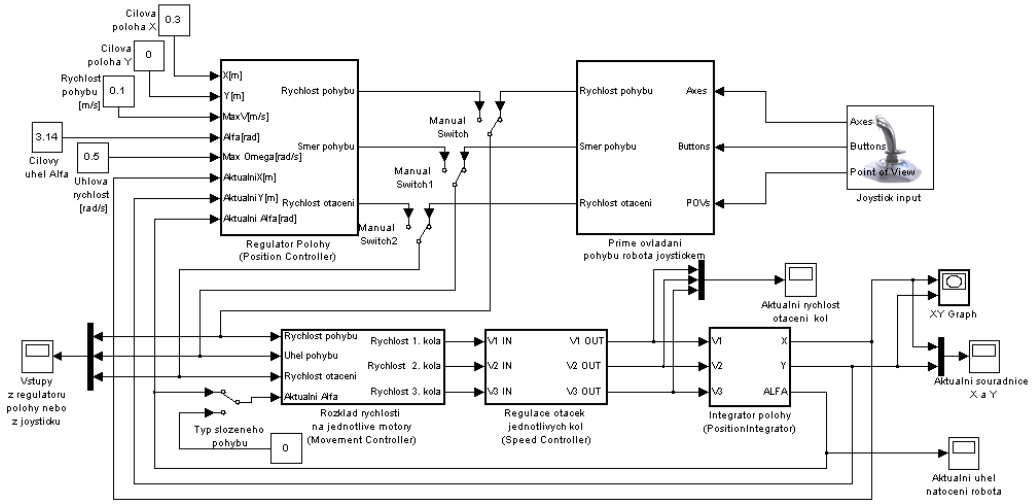
Příloha 1

Úplný seznam parametrů servomotorů Dynamixel

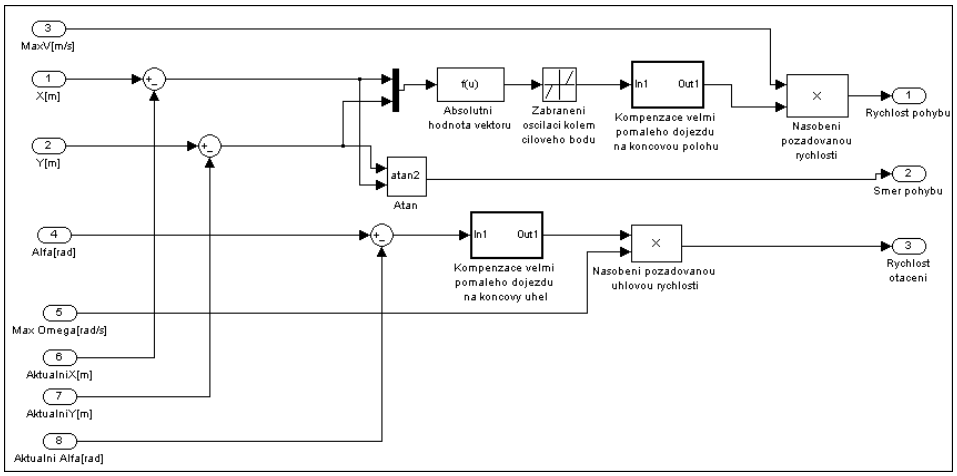
	Address (hexadecimal)	Name	Description	Access	Initial Value (Hexadecimal)
EEPROM Area	0 (0X00)	Model Number(L)	Lowest byte of model number	R	64 (0X40)
	1 (0X01)	Model Number(H)	Highest byte of model number	R	0 (0X00)
	2 (0X02)	Version of Firmware	Information on the version of firmware	R	-
	3 (0X03)	ID	ID of Dynamixel	RW	1 (0X01)
	4 (0X04)	Baud Rate	Baud Rate of Dynamixel	RW	34 (0X22)
	5 (0X05]	Return Delay Time	Return Delay Time	RW	250 (0XFA)
	6 (0X06)	CW Angle Limit(L)	Lowest byte of clockwise Angle Limit	RW	0 (0X00)
	7 (0X07)	CW Angle Limit(H)	Highest byte of clockwise Angle Limit	RW	0 (0X00)
	8 (0X08)	CCW Angle Limit(L)	Lowest byte of counterclockwise Angle Limit	RW	255 (0XFF)
	9 (0X09)	CCW Angle Limit(H)	Highest byte of counterclockwise Angle Limit	RW	3 (0X03)
	11 (0X0B)	the Highest Limit Temperature	Internal Limit Temperature	RW	80 (0X50)
	12 (0X0C)	the Lowest Limit Voltage	Lowest Limit Voltage	RW	60 (0X3C)
	13 [0X0D)	the Highest Limit Voltage	Highest Limit Voltage	RW	240 (0XF0)
	14 (0X0E)	Max Torque(L)	Lowest byte of Max. Torque	RW	255 (0XFF)
	15 (0X0F)	Max Torque(H)	Highest byte of Max. Torque	RW	3 (0X03)
	16 (0X10)	Status Return Level	Status Return Level	RW	2 (0X02)
	17 (0X11)	Alarm LED	LED for Alarm	RW	36 (0X24)
	18 (0X12)	Alarm Shutdown	Shutdown for Alarm	RW	36 (0X24)
RAM Area	24 (0X18)	Torque Enable	Torque On/Off	RW	0 (0X00)
	25 (0X19)	LED	LED On/Off	RW	0 (0X00)
	26 (0X1A)	CW Compliance Margin	CW Compliance margin	RW	0 (0X00)
	27 (0X1B)	CCW Compliance Margin	CCW Compliance margin	RW	0 (0X00)
	28 (0X1C)	CW Compliance Slope	CW Compliance slope	RW	32 (0X20)
	29 (0X1D)	CCW Compliance Slope	CCW Comliance slope	RW	32 (0X20)
	30 (0X1E)	Goal Position(L)	Lowest byte of Goal Position	RW	-
	31 (0X1F)	Goal Position(H)	Highest byte of Goal Position	RW	-
	32 (0X20)	Moving Speed(L)	Lowest byte of Moving Speed	RW	-
	33 (0X21)	Moving Speed(H)	Highest byte of Moving Speed	RW	-
	34 (0X22)	Torque Limit(L)	Lowest byte of Torque Limit	RW	ADD14
	35 (0X23)	Torque Limit(H)	Highest byte of Torque Limit	RW	ADD15
	36 (0X24)	Present Position(L)	Lowest byte of Current Position	R	-
	37 (0X25)	Present Position(H)	Highest byte of Current Position	R	-
	38 (0X26)	Present Speed(L)	Lowest byte of Current Speed	R	-
	39 (0X27)	Present Speed(H)	Highest byte of Current Speed	R	-
	40 (0X28)	Present Load(L)	Lowest byte of Current Load	R	-
	41 (0X29)	Present Load(H)	Highest byte of Current Load	R	-
	42 (0X2A)	Present Voltage	Current Voltage	R	-
	43 (0X2B)	Present Temperature	Current Temperature	R	-
	44 (0X2C)	Registered Instruction	Means if Instruction is registered	RW	0 (0X00)
	46 (0X2E)	Moving	Means if there is any movement	R	0 (0X00)
	47 (0X2F)	Lock	Locking EEPROM	RW	0 (0X00)
	48 (0X30)	Punch(L)	Lowest byte of Punch	RW	32 (0X20)
	49 (0X31)	Punch(H)	Highest byte of Punch	RW	0 (0X00)

Příloha 2

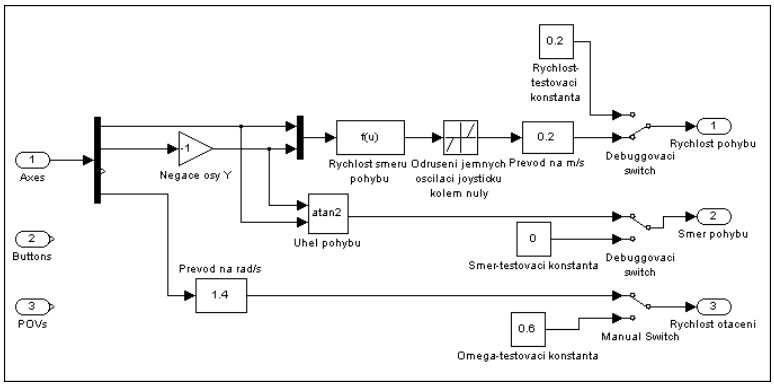
Simulační schéma



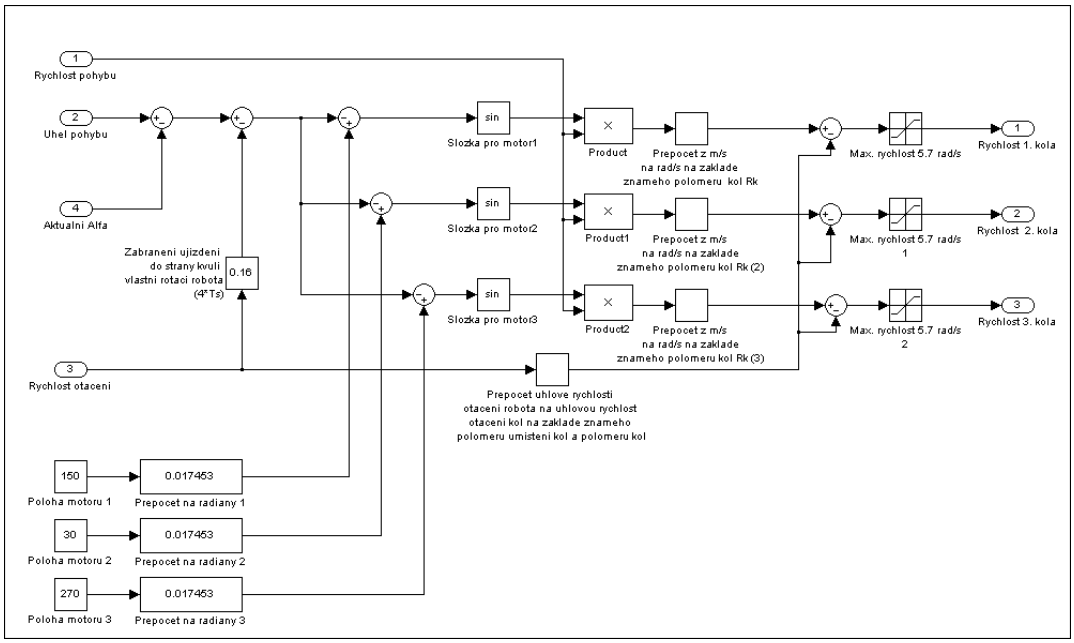
Regulátor polohy (třída PositionController)



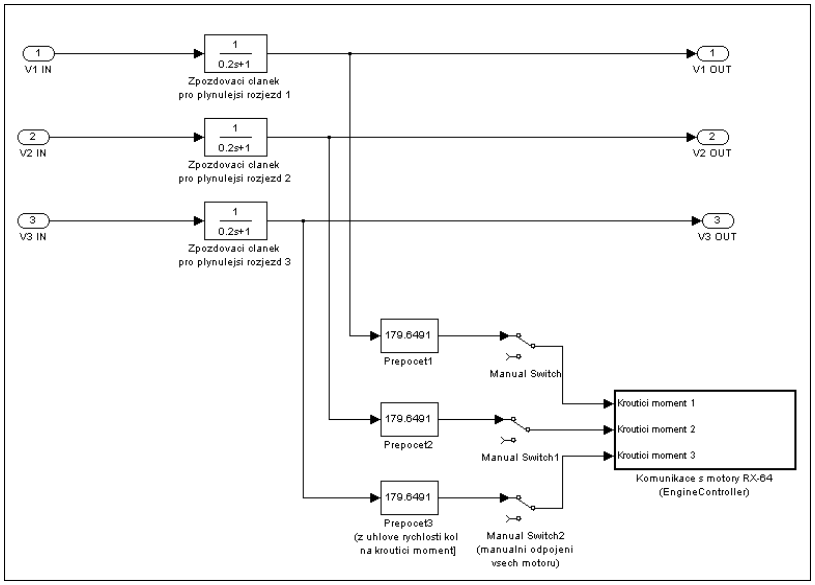
Přímé ovládání pohybu robota joystickem (třída Joystick Controller)



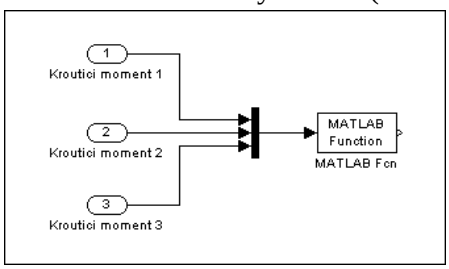
Rozklad rychlosti na jednotlivé motory (třída MovementController)



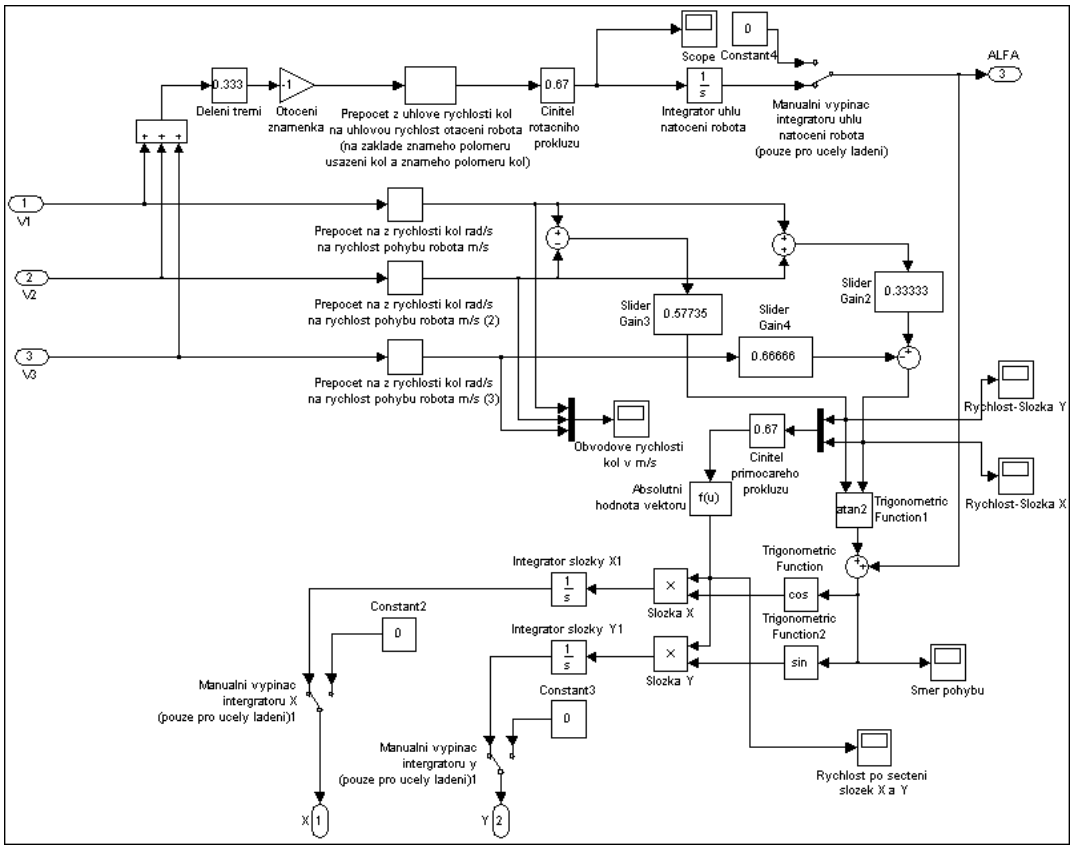
Ovládání otáček jednotlivých kol (třída SpeedController)



Komunikace s motory RX-64 (třída EngineController)

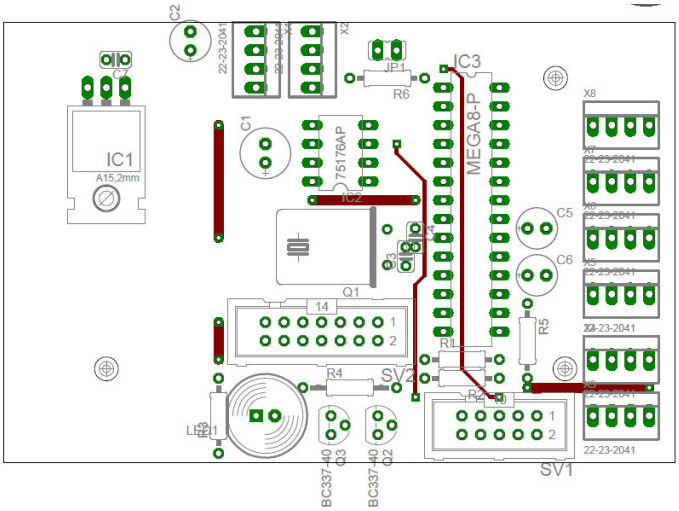


Integrátor polohy (třída Position Integrator)



Příloha 3

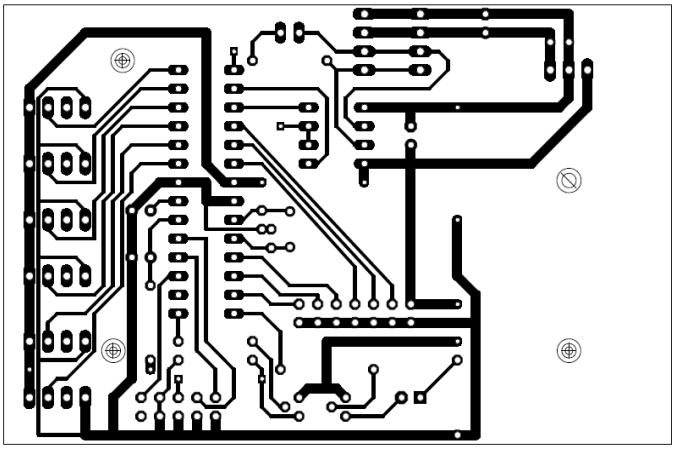
Osazení součástek senzorické desky



Seznam součástek

mikrokontroler Atmel AVR Mega8
integrovaný stabilizátor 7805 s chladičem
75176 - obousměrný budič RS485
T1, T2 – tranzistor BC337-40
R1- rezistor 3k3 0.6W
R2,R4- rezistor 10k 0.6W
R3 –rezistor 470R 0.6W
R5 –rezistor 68R 0.6W
R6 –rezistor 120R 0.6W
LED1 - červená LED průměr 10mm
C1,C2 -elektrol. kondenzátor 220uF/35V
C3,C4 -kondenzátor 22pF
C5,C6 –kondenzátor 1uF/50V
C7 – kondenzátor 100nF
X1-X8 – 8x konektor PSH02-04P
SV1 -konektor MLW10G
SV2 – konektor MLW14G
Q1 -krystal 11.052 MHz

Návrh plošného spoje



Příloha 4

Obsah CD

Eagle	– adresář obsajující schéma senzorické desky a návrh plošného spoje
Firmware	– adresář se zdrojovými kódy pro mikrokontroler Atmel AVR Mega8
Literatura	– adresář s pdf dokumenty citovanými v seznamu literatury
Matlab	– adresář se simulačními schématy MatlabSimulink a knihovnou pro komunikaci se servomotory Dynamixel
Software	– adresář se zdrojovými kódy pro řídící software běžící na PC
Bakalářská práce.pdf	– elektronická verze dokumentu